Appunti di Calcolo Numerico

del corso di Laurea e Diploma in Informatica

Indice

Capitolo 1

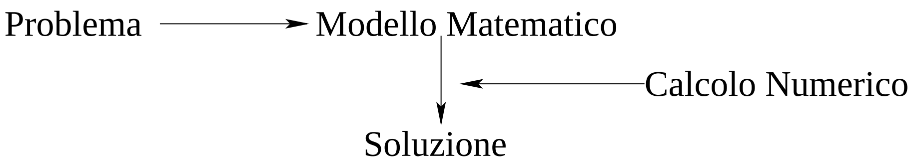
Errori

1.1 Introduzione

Il calcolo numerico è una disciplina che nasce con l'informatica in quanto è strettamente legata al calcolatore. Solitamente si ha un problema da risolvere per il quale è necessario trovare un *metodo numerico* appropriato il quale porterà alla soluzione del suddetto problema. Per metodi numerici s'intendono delle metodologie di soluzione per determinate classi di problemi matematici, i requisiti sono essenzialmente due:

- *Accuratezza*
- implementazione (su di un calcolatore) facile

Si parla di accuratezza perchè ci imbattiamo in alcune tipologie di errore e quindi dobbiamo tener conto delle differenza tra la soluzione esatta e quella fornita dal metodo. Il calcolo numerico si piazza a met‘a tra il modello matematico e la soluzione come evidenzia la seguente figura



1.2 Tipi di errore che possiamo incontrare

Essenzialmente ci possiamo imbattere nei seguenti tipi di errore:

1. Errori di *discretizzazione*
2. Errori di *troncamento*

3. Errori di *arrotindamento* o di *round-off round-on*

vediamoli in dettaglio

1.2.1 Errori di discretizzazione

E'insito nella formulazione del metodo, in quanto noi abbiamo un problema sul *continuo* e dobbiamo implementare un metodo sul calcolatore il quale manipolà solo quantitá *discrete* e quindi si ha un errore "portando" sul discreto un problema sul continuo.

Ad esempio sia $f(x)$ derivabile in $x = \overline{x}$, il nostro problema è quello di calcolare $f'(\overline{x})$, la derivazione *simbolica* è un'operazione molto complessa e non rientra nel calcolo numerico, quindi cerchiamo di approssimare $f'(\overline{x})$

$$f'(\overline{x}) = \lim_{h \to 0} \frac{f(\overline{x}+h) - f(\overline{x})}{h}$$

ancora una volta ci troviamo di fronte ad un problema di difficile risoluzione: il calcolo formale del limite e quindi invece di calcolare il limite vero e proprio utilizziamo un'approssimazione di h, $\overline{h} \approx 0$ e calcoliamo

$$f'(\overline{x}) = \frac{f(\overline{x}+\overline{h}) - f(\overline{x})}{\overline{h}}$$

e quindi abbiamo un errore di discretizzazione.

Esempio sia $f(x) = x^2$, $\overline{x} = 1$, $\overline{h} = 0.1$, abbiamo che $\frac{d}{dx}x^2\big|_{x=1} = 2$ mentre la nostra approssimazione vale

$$\frac{(1+0.1)^2 - 1}{0.1} = 2.1$$

1.2.2 Errori di troncamento

Spesso un metodo numerico si basa su una procedura di tipo ierativo e questo come vedremo tra poco comporta qualche problema in quanto non saremo in grado di trovare la soluzione esatta $\overline{x}$, ma una sua approssimazione x_0, ad esempio sia

$$\begin{array}{ccl} x & \approx & x_0 \text{ (nota)} \\ x_1 & = & \Phi(x_0 \\ \cdots & \cdots & \cdots \\ x_{n+1} & = & \Phi(x_n \end{array}$$

dove Φ rappresenta la variabile di iterazione. Dunque il metodo è uan "macchinetta" ricorsiva che ad ogni iterazione approssima la soluzione $\overline{x}$, il metodo si dice convergente se $\lim_{n\to\infty} x_n = \overline{x}$, però questa convergenza è di tipo asintotico e quindi si avrà un'accuratezza infinita nel risultato dopo un numero **infinito** di iterazioni, cosa che assolutamente non ci possiamo permettere. Quello che ci serve è un numero N tale che x_N sia una buona (per noi) approssimazione di $\overline{x}$ e quindi $x_N - \overline{x}$ è l'errore di troncamento che commettiamo.

Ad esempio sia $\alpha > 0$ e $\overline{x} = \sqrt{\alpha}$ e sia $x_0 = \alpha \cdots x_{n+1} = (x_n + \frac{\alpha}{x_n})$, è noto che $\lim_{n \to \infty} x_n = \sqrt{\alpha}$ e ad esempio se scegliamo $N = 4$ abbiamo che $x_N \approx 2.\underbrace{0 \cdots 0}_{7}92 \cdots$

1.2.3 Errori di arrotindamento o di *round-off round-on*

Questo è l'unico tipo di errore del quale non possiamo fare a meno in quanto è possibile evitare errori di discretizzazione se il nostro problema è sul discreto e possiamo evitare errori di troncamento se il metodo "converge presto" o se il metodo non è del tipo indicato sopra; non è peró possibile far niente per togliere errori di arrotondamento poiché sono dovuti alla struttura interna del calcolatore.
Prima di cominciare è necessario introdurre due concetti fondamentali: quello di *errore relativo* e quello di *errore assoluto* . Sia x la soluzione di un problema e $\tilde{x}$ la soluzione approssimata fornita dal metodo in luogo di x allora

$$\Delta_x = \tilde{x} - x \tag{1.1}$$

è l'errore **assoluto** che commettiamo.
Domanda: un errore di 10^{-20} è grande o piccolo?
Risposta : Dipende.
Ad esempio se $x = 1$ allora 10^{-20} è un errore piccolo ma se $x = 10^{-30}$ allora l'errore è grande, quindi l'errore assoluto non è un'informazione sufficiente per dire se stiamo avendo una buona approssimazione del risultato o meno, quindi abbiamo bisogno di un altro tipo di errore, l'errore **relativo** che è una certa quantità normalizzata:

$$\epsilon_x = \frac{\Delta_x}{x} \tag{1.2}$$

dunque

$$\epsilon_x = \frac{\Delta_x}{x} = (x \neq 0) = \frac{\tilde{x} - x}{x} \to \tilde{x} = x(1 + \epsilon_x)$$

se $\epsilon_x \approx 0$ allora l'errore è piccolo, mentre se $|\epsilon|_x \approx 1$ non abbiamo alcuna informazione su x nota $\tilde{x}$ poiché l'errore commesso è troppo elevato.

Esempio: $\begin{cases} x & = & 1000 \\ \tilde{x} & = & 1001 \\ \Delta_x & = & 1 \\ \epsilon_x & = & 10^{-3} \end{cases}$

considero $-\log_{10} |\epsilon_x| = 3$ ed osservo la seguente situazione:

x	1	0	0	0
$\tilde{x}$	1	0	0	1
Numero cifra:	1	2	3	4

si nota che i due valori sono identici fino alla terza cifra ed il numero 3 è lo stesso datoci dal logaritmo, quindi possiamo dire che l'errore se esiste è a partire dalla cifra data da $-\log_b |\epsilon_x|$ per qualsiasi base b.

Ritorniamo agli errori di arrotondamento con uno sguardo alla rappresentazione dei numeri da parte di un calcolatore; ogni lingauggio di programmazione che si rispetti mette a disposizione almeno due tipi di dato:

- Tipo intero
- Tipo reale

vediamo in dettaglio il primo dei due.
In ogni calcolatore moderno il tipo intero viene comunemente espresso in una notazione posizionale, ad esempio 123 in base 10 (o $(123)_{10}$) significa $1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$, ogni numero viene quindi rappresentato come una stringa $\alpha_1 \dots \alpha_n$ con $\alpha_i \in \{0, 1, \dots, 9\}$. In generale se $b \in N$ allora $(\alpha_1, \dots, \alpha_s)_b = \alpha_1 b^{s-1} + \dots, +\alpha_s b^0$ dove $\alpha_i \in \{0, 1, \dots, b-1\}$. Spesso viene aggiunto anche un altro simbolo $\alpha_0 \in \{0, 1\}$ che rappresenta il segno ed il calcolatore è in grado di rappresentare *senza errori* tutti i numeri interi nel *range* $[-\underbrace{(b-1)\dots(b-1)}_{s}, \underbrace{(b-1)\dots(b-1)}_{s}]$ dove il primo numero è $-(b^s - 1)$ ed il secondo $(b^s - 1)$, infatti

$$\underbrace{(b-1)\dots(b-1)}_{s} = (b-1)b^{s-1} + \cdots + (b-1)b^0 = (b-1)\sum_{i=0}^{i=s} b^i =$$

$$= (b-1)\frac{b^s - 1}{b - 1} = b^s - 1$$

Ad esempio se $b = 2$, $s = 15$ allora possiamo rappresentare senza errore tutti i numeri interi nel *range* [-32767,32767].

Vediamo adesso il *tipo reale*, un numero reale viene rappresentato dal calcolatore come una stringa $\alpha_1 \dots \alpha_s \beta_1 \dots \beta_e$ con $\alpha_i, \beta_j \in \{0, 1, \dots, b-1\}$ $\forall i = 1 \dots s, j = 1 \dots e, b \in N$ con $\alpha_1 \neq 0$ perché il numero è in forma normalizzata e quindi la stringa indicata precedentemente identifica il numero

$$0.\underbrace{\alpha_1 \dots \alpha_s}_{mantissa} \cdot b^{\overbrace{\beta_1 \dots \beta_e}^{esponente}} \tag{1.3}$$

come al solito esistono altri due numeri α_0, β_0 che rappresentano il segno della mantissa e dell'esponente. Da notare che

$$b^{-1} \leq \underbrace{\alpha_1 b^{-1} + \cdots + \alpha_s b^{-s}}_{\alpha_1 \dots \alpha_s} < 1 \tag{1.4}$$

Considero il seguente insieme $\mathcal{M} = \{x \in R : x$ ammette una rappresentazione del tipo (1.3) con l'aggiunta di α_0 e $\beta_0\}$, questo insueme viene detto insieme dei *numeri di macchina* e sicuramente è finito in quanto ogni numero di $\mathcal{M}$ è compreso tra questi due numeri:

$$-1 \cdot b^{\overbrace{(b-1)\dots(b-1)}^{e}} \tag{1.5}$$

$$1 \cdot b^{\overbrace{(b-1)\dots(b-1)}^{e}} \tag{1.6}$$

che indicano (all'incirca) il più piccolo ed il più grande numero in virgola mobile rappresentabile dal calcolatore. Lo standard `IEEE754` indica $b = 2$, $s = 52$, 2 segni, $e = 10$ e quindi i numeri rappresentabili dal calcolatore cadono nel *range* $[-10^{308}, 10^{308}]$.

Problema R è denso, $\mathcal{M}$ no, ad esempio il più piccolo numero in modulo di $\mathcal{M}$ è

$$\approx 1 \cdot b^{-\overbrace{(b-1)(b-1)\dots(b-1)}^{e}} \tag{1.7}$$

quindi non potendo lavorare "direttamente" su R si utilizza una funzione $R \to \mathcal{M}$, anzi per essere precisi da $I \subset R \to \mathcal{M}$ in quanto $\mathcal{M}$ oltre a non essere denso è anche limitato, I è l'intervallo delimitato dai due estremi indicati in (1.5) e (1.6), quindi abbiamo bisogno di di

$$\mathrm{fl} : I \to \mathcal{M}$$

se $x \in I$, $\tilde{x} \in \mathcal{M}$ allora $\mathrm{fl}(x) = \tilde{x}$ e si dice che $\tilde{x}$ è il *floating* di x. Come di definisce questa *floating*?
$x = 0.\alpha_1\alpha_2\dots\alpha_s \underbrace{\alpha_{s+1}\dots\dots}_{\text{cifre extra}} b^{\beta_1\dots\beta_e}$ dove l'esponente della b fa rientrare x nel *range* dei numeri di macchina. L'immediata rappresentazione per troncamento è $\mathrm{fl}(x) = 0.\alpha_1\alpha_2\dots\alpha_s b^{\beta_1\dots\beta_e}$, esiste anche un altro tipo di rappresentazione: quella per *arrotondamento* la quale, supponendo b pari (come del accade sempre), opera nel seguente modo: $\mathrm{fl}(x) = 0.\alpha_1\alpha_2\dots\alpha_{\tilde{s}} b^{\beta_1\dots\beta_e}$

$$a_{\tilde{s}} = \begin{cases} \alpha_s + 1 & \text{se } \alpha_{s+1} \geq b/2 \\ \alpha_s & \text{altrimenti} \end{cases}$$

Maggioriamo l'errore relativo nel caso della rappresentazione per troncamento

$$|\epsilon_x| = \frac{|\tilde{x} - x|}{|x|} =$$

$$= \frac{|0.\alpha_1\alpha_2\dots\alpha_s - 0.\alpha_1\alpha_2\dots\alpha_s\alpha_{s+1}\dots\dots|b^{\beta_1\dots\beta_e}}{|0.\alpha_1\alpha_2\dots\alpha_s|b^{\beta_1\dots\beta_e}} =$$

$$= \frac{|0.\overbrace{0\ldots0}^{s}\alpha_{s+1}\ldots|}{|0.\alpha_1\alpha_2\ldots\alpha_s|} = \frac{b^{-s}|0.\alpha_{s+1}\ldots|}{|0.\alpha_1\alpha_2\ldots\alpha_s|} = \text{(tolgo il valore assoluto)} =$$

$$= \frac{b^{-s}(0.\alpha_{s+1}\ldots)}{0.\alpha_1\alpha_2\ldots\alpha_s}$$

considero che $\begin{cases} 0.\alpha_{s+1}\ldots < 1 \\ 0.\alpha_1\alpha_2\ldots\alpha_s \geq b^{-1} \end{cases}$ quindi

$$|\epsilon_x| < \frac{b^{-s}\cdot 1}{b^{-1}} = \frac{b^{-s}}{b^{-1}} = b^{1-s}$$

e quest'ultimo numero prende il nome di *precisione di macchina* **u**.

1.3 Overflow e underflow

Problema: $\forall x \in \mathcal{M}$ vale la seguente disuguaglianza

$$1 \cdot b^{\underbrace{-(b-1)\ldots(b-1)}_{e}} < |x| < 1 \cdot b^{\underbrace{(b-1)\ldots(b-1)}_{e}}$$

se $|x| < 1 \cdot b^{\underbrace{-(b-1)\ldots(b-1)}_{e}}$ si ha *underflow*, mentre se $|x| > 1 \cdot b^{\underbrace{(b-1)\ldots(b-1)}_{e}}$ si ha *overflow*, in entrambi i casi si genera una condizione di errore.

1.3.1 Underflow

Sia $x = 0.10\ldots0 \cdot b^{\beta_1\ldots\beta_e}$ cosa accade nell'eseguire x/b?. Teoricamente andrebbe decrementato l'esponente, ma in pratica non è possibile perché l'esponente è quello minimo rappresentabile, quindi si ricorre ad un aritficio: la mantissa viene "*shiftata*" a destra in modo che $\frac{x}{b} = 0.010\ldots0 \cdot b^{\beta_1\ldots\beta_e}$ ma:

- il numero non è normalizzato
- al massimo è fattibile per s volte

come ultima *recovery* viene posto $\frac{x}{b} = 0$.

1.3.2 Overflow

Se $x = 1 \cdot b^{\underbrace{(b-1)\ldots(b-1)}_{e}}$ moltiplicando o sommando per qualsiasi quantità positiva non vi è alcuna possibilità di *recovery* in quanto sia l'esponente che la mantissa sono massimi e quindi si prospettano due alternative:

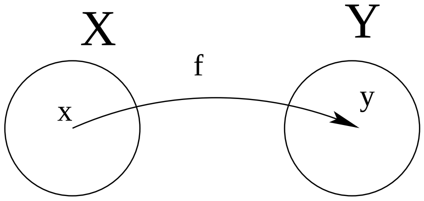
- generare un'eccezione aritmetica
- prevedere una quantità **infinita**

nell'ultimo caso l'analisi ci ricorda di stare attenti a casi del tipo ∞/∞.

1.4 Studio del condizionamento del problema

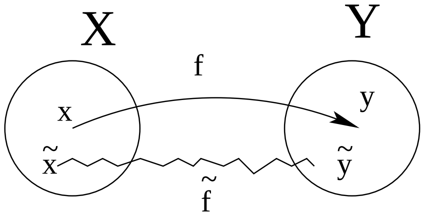
Noi abbiamo $\mathrm{fl}(x) = x(1 + \epsilon_x)$ con $|\epsilon_x| <$**u** quindi l'errore di approssimazione è un fedele compagmo di chi fa uso di un calcolatore. Sia $y = f(x)$ il nostro metodo numerico con

x	dati in ingresso
y	risultato
$f(\cdot)$	forma funzionale del metodo

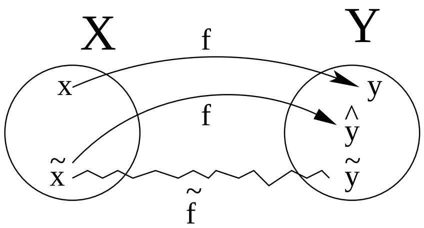


Questo a livello teorico, ma a livello pratico entrano in gioco i vari errori e quindi la situazione passa da $y = f(x)$ a $\tilde{y} = \tilde{f}(\tilde{x})$ con

$\tilde{x}$	dati perturbati (sempre)
$\tilde{y}$	risultato pertutbato
$\tilde{f}(\cdot)$	diversa da $f(\cdot)$ in quanto opera sul discreto, non sul continuo



Stimare $\tilde{f}$ è assai complesso quindi si studia un problema intermedio più semplice: partiamo da $\tilde{x}$ perturbato e consideriamo $f(\tilde{x}) = \hat{y}$, assumere la f equivale ad assumere che i calcoli siano svolti in maniera *esatta*, abbiamo quindi la seguente rappresentazione grafica:



se l'errore che nell'approssimare x con $\tilde{x}$ è dello stesso ordine di quello che c'è tra y e $\tilde{y}$ allora il problema è ben condizionato, altrimenti è mal condizionato; se un problema è ben condizionato è possibile che andando a lavorare sul discreto si ottengano risultati soddisfacenti.

Troviamo una relazione tra $\hat{y}$ e $\tilde{x}$ o meglio tra ϵ_y e ϵ_x

$$\hat{y} = y(1+\epsilon_y) = f(\tilde{x}) = f(x(1+\epsilon_x))$$

assumiamo che **non avvengano errori** nel calcolo della f qundi dato che $y = f(x)$ l'uguaglianza precedente si riscrive in

$$y\epsilon_y = f'(x)x\epsilon_x + o(|x\epsilon_x|^2)$$

dove $f(x+x\epsilon_x)$ viene approssimato con la formula di Taylor in $f(x)+f'(x)x\epsilon_x + o(|x\epsilon_x|^2)$, $o(|x\epsilon_x|^2)$ viene trascurato e quindi l'uguaglianza si riscrive in $y\epsilon_y = f'(x)x\epsilon_x$ quindi $|\epsilon_y| \leq |f'(x)x/y||\epsilon_x|$ noi indicheremo con k il numero $|f'(x)x/y|$ detto numero di condizionamento del problema.

$|\epsilon_y| \leq k|\epsilon_x||$ dove k mi da una stima dei dati che posso ottenere partendo da una x perturbata, se $k \approx 1$ allora $|\epsilon_y| \approx |\epsilon_x|$ ed il problema è ben condizionato se $k \gg 1$ il problema è mal condizionato, da un aperturbazione piccola in ingresso si ottiene una perturbazione troppo grande dei dati in uscita.

Fino a che valori di k il problema è da prendere in considerazione? Ad esempio $k = 1$,$k = 2$,... o quale altro valore? Sicuramente $|\epsilon_y| = k|\epsilon_x| < k\mathbf{u} < 1$ allora $k < 1/\mathbf{u}$ altrimenti i valori ottenuti sono senza senso.

♡

Capitolo 2

Sistemi Lineari

2.1 Introduzione

Consideriamo sistemi lineari di n equazioni in n incognite.

In generale un sistema di n equazioni in n incognite è della forma

$$\begin{cases} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ \vdots & & & & & & & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{cases} \tag{2.1}$$

In questo caso la matrice dei coefficienti $A = (a_{ij})_{i,j=1..n}$ è una matrice quadrata $n \times n$ e, definendo $\underline{x} = (x_1 \cdots x_n)^T$ e $\underline{b} = (b_1 \cdots b_n)^T$, il sistema può essere scritto nella forma

$$A\underline{x} = \underline{b} \tag{2.2}$$

Se A è non singolare, per il teorema di Rouché-Capelli, la soluzione del sistema è unica e si può scrivere come

$$\underline{x} = A^{-1}\underline{b} \tag{2.3}$$

In pratica però non si usa quasi mai la (2.3) per la risoluzione del sistema, perché il calcolo della matrice inversa è molto complicato.

2.2 Casi particolari di sistemi lineari facili da risolvere

2.2.1 Matrice dei coefficienti diagonale

Se la matrice dei coefficienti è diagonale cioè è della forma

$$\mathbf{A} = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & d_n \end{pmatrix} \tag{2.4}$$

e quindi il sistema è

$$\begin{cases} d_1 x_1 & = & b_1 \\ d_2 x_2 & = & b_2 \\ \vdots & & \vdots \\ d_n x_n & = & b_n \end{cases}$$

allora si ha $\det(A) = \prod_{i=1}^{n} d_i$ quindi $\det(A) \neq 0 \iff d_i \neq 0 \forall i = 1..n$. Le soluzioni sono evidentemente $x_i = b_i/di$ per $i = 1..n$. Quindi *il costo della risoluzione di un sistema la cui matrice dei coefficienti è diagonale è n flops (n divisioni)*

2.2.2 Matrice dei coefficienti triangolare

Se A è una matrice triangolare inferiore (cioè se $a_{ij} = 0$ quando $j > i$)

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & 0 \\ \vdots & \ddots & 0 \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

oppure triangolare superiore ($a_{ij} = 0$ quando $j < i$)

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ 0 & \ddots & \vdots \\ 0 & \cdots & a_{nn} \end{pmatrix}$$

il determinante si può facilmente calcolare come $\prod_{i=1}^{n} a_{ii}$ quindi $\det(A) = 0 \iff a_{ii} = 0 \forall i = 1..n$

In questi casi è possibile trovare le soluzioni con una sostituzione all'avanti o all'indietro. Quindi *il costo della risoluzione di un sistema la cui matrice deei coefficienti è triangolare è dell'ordine di n^2 flops.*

2.2.3 Matrice dei coefficienti ortogonale

Definizione 2.1 *A si dice* ortogonale *se* $A^T A = AA^T = I$.

In questo caso l'inversa è facile da calcolare perché $A^{-1} = A^T$ quindi le soluzioni del sistema si ottengono con la formula

$$\underline{x} = A^{-1}\underline{b} = A^T\underline{b}$$

Il prodotto scalare di una matrice $n \times n$ per un vettore richiede n prodotti scalari, ognuno dei quali richiede $2n$ operazioni elementari quindi *il costo della risoluzione di un sistema lineare la cui matrice dei coefficienti è ortogonale è pari a* $n * (2n) = O(n^2)$ *flops.*

2.3 La fattorizzazione

Quando la matrice dei coefficienti non appartiene a nessuna delle classi viste nel paragrafo precedente, per risolvere il sistema si ricorre alla fattorizzazione della matrice, cioè si cercano due matrici F_1 e F_2, appartenenti a due classi di matrici per le quali sappiamo risolvere il corrispondente sistema, tali che $A = F_1F_2$.

Vederemo due tipi di fattorizzazione: la fattorizzazione LU e la fattorizzazione QR.

2.4 Proprietà delle matrici triangolari

Per continuare ci servono alcune proprietà delle matrici diagonali.

1. Se A_1 e A_2 sono matrici triangolari inferiori (risp. superiori) allora il loro prodotto A_1A_2 è una matrice triangolare inferiore (risp. superiore)

2. Se A_1 e A_2 sono matrici triangolari inferiori (risp. superiori) con gli elementi diagonali uguali a 1 allora il loro prodotto A_1A_2 è una matrice triangolare inferiore (risp. superiore) con gli elementi diagonali uguali a 1

3. Se A è una matrice triangolare inferiore (risp. superiore) e non singolare allora A^{-1} è una matrice triangolare inferiore (risp. superiore). Inoltre se A ha gli elementi diagonali uguali a 1 allora anche gli elementi diagonali di A^{-1} sono 1.

2.5 La fattorizzazione LU

2.5.1 Presentazione del metodo

Il metodo della fattorizzazione LU consiste nel cercare di scrivere

$$A = LU$$

dove

- $L = (l_{ij})$ è una matrice triangolare inferiore con elementi diagonali uguali a 1 ($l_{ii} = 1$ per $i = 1..n$, $l_{ij} = 0$ per $j > i$). Notiamo che questa condizione implica $\det(L) = 1$.
- $U = (u_{ij})$ è una matrice triangolare superiore ($u_{ij} = 0$ se $j < i$)

Se questa fattorizzazione esiste, A si dice *fattorizzabile LU*.

2.5.2 Unicità della fattorizzazione LU

Teorema 2.1 *Se A è fattorizzabile LU, questa fattorizzazione è unica.*

Dimostrazione: Supponiamo, per assurdo, che esistano due fattorizzazioni diverse per A:

$$A = L_1U_1 = L_2U_2 \tag{2.5}$$

dove L_1, U_1, L_2, U_2 soddisfano le proprietà della fattorizzazione LU. Allora si ha:

$$0 \neq \det(A) = \det(L_1U_1) = \underbrace{\det(L_1)}_{1}\det(U_1) = \det(U_1) \tag{2.6}$$

La (2.6) vale anche se sostituiamo U_1 con U_2 e L_1 con L_2, inoltre si ha $\det(L_1) = \det(L_2) = 1$ quindi i fattori L_1, L_2, U_1, U_2 sono matrici non singolari e ammettono la matrice inversa.

$$\begin{aligned} L_1U_1 &= L_2U_2 \\ L_1 &= L_2U_2U_1^{-1} \\ L_2^{-1}L_1 &= U_2U_1^{-1} \end{aligned} \tag{2.7}$$

Per le proprietà enunciate nella sezione 2.4, $L_2^{-1}L_1$ è una matrice triangolare inferiore mentre $U_2U_1^{-1}$ è una matrice triangolare superiore. Perché queste due matrici siano uguali devono essere quindi diagonali. Però poiché L_1 e L_2 hanno gli elementi della diagonale principali uguali a 1, si ha che

$$L_2^{-1}L_1 = U_2U_1^{-1} = I \tag{2.8}$$

Dalla (2.8) segue immediatamente che $L_1 = L_2$ e $U_1 = U_2$ cvd.

2.5.3 Algoritmo di fattorizzazione LU

In $n-1$ passi si trasforma $A \in \mathbf{R}^{n\times n}$ in una matrice triangolare superiore azzerando via via gli elementi di ogni colonna al di sotto della diagonale. Per far questo si moltiplica A per delle matrici $L_1, L_2, \cdots, L_{n-1}$.

Indichiamo con $A^{(k)}$ la matrice al passo k-esimo. Aggiungiamo ad ogni elemento della matrice un indice superiore che indica in quale passo è stato modificato per l'ultima volta.

Primo Passo

Perché questo passo sia possibile è neccessario imporre $a_{11}^{(1)} \neq 0$.
Definiamo il primo *vettore di Gauss*

$$\underline{g}_1 = \frac{1}{a_{11}^{(1)}} \begin{pmatrix} 0 & a_{21}^{(1)} & \cdots & a_{n1}^{(1)} \end{pmatrix}^T \tag{2.9}$$

e la prima *matrice elementare di Gauss*

$$L_1 = I - \underline{g}_1 \underline{e}_1^T \tag{2.10}$$

Notiamo che nella (2.9), l'indice di $\underline{g}$ è 1 e il suo primo elemento è nullo.
La matrice A dopo il primo passo diventa

$$A^{(2)} = L_1 A = (I - \underline{g}_1 \underline{e}_1^T)A = A - \underline{g}_1 (e_1^T A) \tag{2.11}$$

Poiché il primo elemento di $\underline{g}_1$ è 0 il prodotto $\underline{g}_1(e_1^T A)$ non modifica gli elementi della prima riga di A.

La prima colonna di $A^{(2)}$ può essere ricavata in questo modo

$$\begin{aligned} A^{(2)}\underline{e}_1 &= L_1 A\underline{e}_1 = A\underline{e}_1 - \underline{g}_1 \underbrace{e_1^T A \underline{e}_1}_{a_{11}^{(1)}} = \\ &= A\underline{e_1} - \begin{pmatrix} 0 & a_{21}^{(1)} & \cdots & a_{n1}^{(1)} \end{pmatrix}^T = \\ &= \begin{pmatrix} a_{11}^{(1)} \\ a_{21}^{(1)} \\ \vdots \\ a_{n1}^{(1)} \end{pmatrix} - \begin{pmatrix} 0 \\ a_{21}^{(1)} \\ \vdots \\ a_{n1}^{(1)} \end{pmatrix} = \begin{pmatrix} a_{11}^{(1)} \\ 0 \\ \vdots \\ 0 \end{pmatrix} \end{aligned} \tag{2.12}$$

Al primo passo vengono azzerati tutti gli elementi della prima colonna al di sotto di a_{11}. Questo prodotto modifica anche tutti gli elementi delle righe $2..n$.
Alla fine del primo passo otteniamo quindi la matrice

$$\mathbf{A}^{(2)} = \mathbf{L_1 A} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{pmatrix} \tag{2.13}$$

Secondo Passo

Perché questo passo sia possibile dev'essere $a_{22}^{(2)} \neq 0$ Definiamo il secondo *vettore di Gauss*

$$\underline{g}_{(2)} = \frac{1}{a_{22}^{(2)}} \begin{pmatrix} 0 & 0 & a_{32}^{(2)} & \cdots & a_{n2}^{(2)} \end{pmatrix}^T \tag{2.14}$$

e la seconda *matrice elementare di Gauss*

$$L_2 = I - \underline{g}_2 \underline{e}_2^T \tag{2.15}$$

Al secondo passo gli elementi delle prime due righe non vengono modificati. Si ottiene

$$\mathbf{A}^{(3)} = \mathbf{L_2 L_1 A} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & a_{3n}^{(3)} & \cdots & a_{nn}^{(3)} \end{pmatrix} \tag{2.16}$$

Passo i-esimo

Perché questo passo sia possibile dev'essere $a_{ii}^{(i)} \neq 0$ Definiamo l'i-esimo *vettore di Gauss*. Notiamo che $\underline{g}_i$ ha le prime i componenti nulle.

$$\underline{g}_i = \frac{1}{a_{ii}^i} \begin{pmatrix} 0 & \cdots & 0 & a_{(i+1)i}^{(i)} & \cdots & a_{ni}^{(i)} \end{pmatrix}^T \tag{2.17}$$

e la i-esima *matrice elementare di Gauss*

$$L_i = I - \underline{g}_i \underline{e}_i^T \tag{2.18}$$

Con $n-1$ iterazioni abbiamo trasformato la matrice in triangolare superiore. Questa fattorizzazione è stata possibile se $a_{ii}^{(i)} \neq 0$ per $i = 1..n-1$. Alla fine otteniamo quindi $A^{(n)} = U$.

$$\mathbf{U} = \mathbf{A}^{(n)} = \mathbf{L_{n-1}} \cdots \mathbf{L_1 A} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} \end{pmatrix} \tag{2.19}$$

Poichè $L_1, \cdots, L_{n-1}$ sono matrici triangolari inferiori con elementi diagonali uguali a 1 (cfr. sez 2.5.4) e quindi non singolari, possiamo definire

$$\begin{aligned} L^{-1} &= L_{n-1} \cdots L_1 \\ L^{-1}A = U &\Rightarrow A = LU \end{aligned} \tag{2.20}$$

e, per le proprietà enunciate nella sezione 2.4, anche L è una matrice triangolare inferiore con elementi diagonali uguali a 1.

2.5.4 Condizioni di esistenza della fattorizzazione LU

Teorema 2.2 (Condizione di esistenza della fattorizzazione LU (1)) *A è fattorizzabile LU se e solo se $a_{ii}^{(i)} \neq 0$ per $i = 1..n$.*

Dimostrazione: Consideriamo la forma della matrice L_i.

$$\mathbf{L_i} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ * & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & * & 1 \end{pmatrix} \tag{2.21}$$

$L_i = I - \underline{g}_i \underline{e}_i^T$ quindi, poichè $\underline{g}_i$ ha le prime i componenti nulle, i termini diversi da 0 e da 1 (indicati dagli asterischi) si trovano al di sotto della diagonale principale.

L_i è quindi triangolare inferiore con elementi diagonali uguali a 1. Questo implica che $\det(L_i) = 1$ quindi

$$\det(U) = \det(L_{n-1} \cdots L_1 A) = \det(L_1)\det(L_2)\cdots\det(L_1)\det(A) = \det(A) \tag{2.22}$$

Dal primo e dall'ultimo termine della (2.22) si ricava $\det(A) = \det(U)$, ma U è triangolare superiore perciò:

$$\det(U) = \prod_{j=1}^{n} a_{jj}^{(j)} \tag{2.23}$$

Dalla (2.23) segue che se A è non singolare e fattorizzabile LU (e quindi abbiamo imposto $a_{jj}^{(j)} \neq 0$ per $j = 1..n-1$) allora anche $a_{nn}^{(n)} \neq 0$. È così dimostrata la condizione per l'esistenza della fattorizzazione LU

$$a_{ii}^{(i)} \neq 0 \qquad \forall i = 1..n \tag{2.24}$$

Matrice Principale di ordine k, A_k

Definizione 2.2 *Se* $A = (a_{ij}) \in \mathbf{R}^{n\times n}$ *e* $k \leq n$, *allora si dice* matrice principale di ordine k , *indicata con* A_k *la matrice*

$$\mathbf{A_k} = \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix} \tag{2.25}$$

La matrice A_k può essere ottenuta prelevando le prime k righe di A e quindi le prime k colonne della matrice così ottenuta, in questo modo:

$$\mathbf{A_k} = \begin{bmatrix} \mathbf{I_k} & \underline{\mathbf{0}}_{\mathbf{k(n-k)}} \end{bmatrix} \mathbf{A} \begin{bmatrix} \mathbf{I_k} \\ \underline{\mathbf{0}}_{\mathbf{(n-k)k}} \end{bmatrix} \tag{2.26}$$

Teorema 2.3 *Se* A *è fattorizzabile LU allora anche* A_k *lo è.*

Dimostrazione: Per ipotesi si ha $A = LU$, sostituendo nella (2.26) si ottiene

$$\mathbf{A_k} = \left(\begin{bmatrix} \mathbf{I_k} & \underline{\mathbf{0}}_{\mathbf{k(n-k)}} \end{bmatrix} \mathbf{L}\right)\left(\mathbf{U}\begin{bmatrix} \mathbf{I_k} \\ \underline{\mathbf{0}}_{\mathbf{(n-k)k}} \end{bmatrix}\right) =$$

$$\begin{bmatrix} \mathbf{L_k} & \underline{\mathbf{0}}_{\mathbf{k(n-k)}} \end{bmatrix} \begin{bmatrix} \mathbf{U_k} \\ \underline{\mathbf{0}}_{\mathbf{(n-k)k}} \end{bmatrix} = \mathbf{L_k U_k} + \underline{\mathbf{0}}_{\mathbf{kk}} = \mathbf{L_k U_k} \tag{2.27}$$

Abbiamo così dimostrato che se A è fattorizzabile, anche A_k lo è (infatti $A_k = L_k U_k$)

Definizione 2.3 *Si dice* minore principale di ordine k della matrice A_k*,* $\det(A_k)$

Teorema 2.4 (Condizione di esistenza della fattorizzazione LU (2)) *A è fattorizzabile LU se e solo se tutti i minori principali di A sono non nulli*

Dimostrazione:

$$\det(A_k) = \det(L_k U_k) = \underbrace{\det(L_k)}_{1} \det(U_k) = \det(U_k) = \prod_{j=1}^{k} a_{jj}^{(j)} \tag{2.28}$$

L'ultimo termine dell'equazione (2.28) è la condizione d'esistenza della fattorizzazione LU (Teorema 2.2) quindi abbiamo trovato una condizione di esistenza della fattorizzazione LU, verificabile a priori sulla matrice A.

Bisogna notare che questa condizione è più restrittiva della condizione richiesta perché il sistema sia compatibile che è $\det(A) \neq 0$. Per questo motivo, nel seguito, seguiremo due strade che permettono di ottenere algoritmi più generali:

1. Cercare matrici nelle quali la non singolarità implichi la non singolarità di tutte le matrici principali. A queste matrici è conveniente applicare la fattorizzazione LU.

2. Modificare la procedura in modo che chieda solo la non singolarità di A.

2.5.5 Costo della fattorizzazione LU

Occupazione di memoria

I dati di input sono A e b che occupano rispettivamente n^2 e n locazioni di memoria.

Teorema 2.5 *È possibile riscrivere A con le informazioni significative dei suoi fattori L e U.*

Dimostrazione: Ricordiamo che nella (2.20) è stata definita la matrice L come

$$L = (L_{n-1} \cdots L_1)^{-1} = L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} \tag{2.29}$$

dove la i-esima matrice di Gauss, L_i, è definita come

$$L_i = I - \underline{g}_i \underline{e}_i^T \tag{2.30}$$

$$\underline{g}_i = \frac{1}{a_{ii}^i} \begin{pmatrix} 0 & \cdots & 0 & a_{(i+1)i}^{(i)} & \cdots & a_{ni}^{(i)} \end{pmatrix}^T \tag{2.31}$$

Teorema 2.6 $L_i^{-1} = I + \underline{g}_i \underline{e}_i^T$

Dimostrazione:

$$\begin{aligned} L_i L_i^{-1} &= (I - \underline{g}_i \underline{e}_i^T)(I + \underline{g}_i \underline{e}_i^T) = \\ &= I + \underline{g}_i \underline{e}_i^T - \underline{g}_i \underline{e}_i^T - \underline{g}_i \underbrace{\underline{e}_i^T \underline{g}_i}_{g_i(i)=0} \underline{e}_i^T = I \end{aligned} \tag{2.32}$$

Abbiamo così dimostrato che $L_i^{-1} = I + \underline{g}_i \underline{e}_i^T$. Sostituendo questa espressione della L_i nella (2.29) si ottiene:

$$\begin{aligned} L &= (I + \underline{g}_1 \underline{e}_1^T)(I + \underline{g}_2 \underline{e}_2^T) \cdots (I + \underline{g}_{n-1} \underline{e}_{n-1}^T) = \\ & I + \underline{g}_1 \underline{e}_1^T + \cdots + \underline{g}_{n-1} \underline{e}_{n-1}^T + O_{i<j}(\underline{g}_i \underbrace{\underline{e}_i^T \underline{g}_j}_{0} \underline{e}_j^T) \end{aligned} \tag{2.33}$$

dove $\underline{e}_i^T \underline{g}_j$ perché $i < j$ e le prime j componenti di $\underline{g}_j$ sono nulle.

$$L = I + \underline{g}_1 \underline{e}_1^T + \cdots + \underline{g}_{n-1} \underline{e}_{n-1}^T = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ g_{21} & \ddots & & 0 \\ \vdots & & \ddots & \\ g_{n1} & g_{n2} & \cdots & 1 \end{pmatrix} \tag{2.34}$$

Di L ci interessa memorizzare solo la parte strettamente triangoolare inferiore, perché gli elementi della diagonale principale sono tutti 1. Osserviamo anche, dalla (2.34), che gli elementi della parte triangolare inferiore di L sono gli elementi significativi (cioè diversi da 0) dei vettori di Gauss. Inoltre dela U ci interessa memorizzare solo la parte triangolare superiore perchè gli altri elementi sono nulli.

Queste considerazioni permettono di concludere che *è possibile riscrivere A memorizzando gli elementi significativi dei vettori di Gauss nella parte strettamente triangolare inferiore, e ottenendo, dopo $n-1$ passi, nella parte triangolare superiore la matrice U.*

Complessità computazionale

Teorema 2.7 *La fattorizzazione LU ha una complessità dell'ordine di $\frac{2}{3}n^3$ flops.*

L'algoritmo esegue $n-1$ passi. Al passo i-esimo vengono eseguite $n-i$ divisioni per calcolare gli elementi significativi del vettore di Gauss e 2 operazioni elementari (una sottrazione e una moltiplicazione) per ognuno degli elementi della sottomatrice che viene modificata (questi elementi sono $(n-i)^2$). Otteniamo quindi la seguente espressione del costo della fattorizzazione LU:

$$\begin{aligned}\sum_{i=1}^{n-1}(n-i)+2(n-i)^2 = \sum_{k=1}^{n-1} k+2k^2 = \\ \frac{(n-1)n}{2}+2\frac{(n-1)n(2n-1)}{6} = O(\frac{2}{3}n^3)\end{aligned} \tag{2.35}$$

2.6 Algoritmo di Gauss con pivoting parziale

Finora abbiamo visto come per risolvere un sistema del tipo

$$\mathbf{A}\underline{x} = \underline{b} \tag{2.36}$$

cerchiamo la sua fattorizzazione LU considerando una matrice A non singolare cioé:

$$\mathbf{A} \in \Re^{(n \times n)} \quad \text{con } det(\mathbf{A}) \neq 0 \tag{2.37}$$

In particolare abbiamo cercato una fattorizzazione LU giungendo all'equazione A=LU che ci permetteva di risolvere il sistema iniziale risolvendo i sistemi:

$$\begin{cases} \mathbf{L}\underline{y} = \underline{b} \\ \mathbf{U}\underline{x} = \underline{y} \end{cases}$$

Abbiamo anche visto che tale fattorizzazione ha un costo dell'ordine di $\frac{2}{3}n^3$ flops. In realtá peró abbiamo costruito un algoritmo che richiede non soltanto la non singolaritá della matrice **A**, ma anche la non singolaritá di tutte le matrici principali di ordine k con $k = 1..n$, infatti abbiamo che

$$\mathbf{A} = \mathbf{LU} \Longleftrightarrow \text{ tutti i minori principali sono non nulli} \tag{2.38}$$

ma tale condizione è molto piú forte della richiesta di esclusiva non singolaritá della matrice **A**. Il nostro scopo sarà quindi quello di ricercare un metodo che garantisca la fattorizzazione della matrice **A** sotto la sola condizione di non singolarità della matrice stessa. Riprendendo l'algoritmo visto in precedenza definiamo:

$$\mathbf{A} = \begin{pmatrix} a_{11}^{(1)} & . & . & . & a_{1n}^{(1)} \\ . & & & & . \\ . & & & & . \\ . & & & & . \\ a_{n1}^{(1)} & . & . & . & a_{nn}^{(1)} \end{pmatrix} = \mathbf{A}^{(1)} \tag{2.39}$$

$$\underline{g}_1 = 1/a_{11}^{(1)} \ [\ 0 \ a_{21}^{(1)} \ ... \ a_{n1}^{(1)} \ ]^T \text{ per cui } a_{11}^{(1)} \neq 0 \tag{2.40}$$

$$\mathbf{L}_1 = \mathbf{I} - \underline{g}_1 \underline{e}_1^T \tag{2.41}$$

con g_1 primo vettore di Gauss e $\underline{e}_1$ prima base canonica. A questo punto se $a_{11}^{(1)} = 0$ l'eliminazione di Gauss terminerebbe subito. Sfruttando peró la non singolaritá della matrice **A** possiamo affermare che sulla prima colonna ci deve essere almeno un elemento $\neq 0$. Definiamo a ora questo elemento:

$$| \ a_{k_1 1}^{(1)} \ | = \max_{i:1..n} | \ a_{i1}^{(1)} \ | \qquad \text{che sará nella riga } k_1$$

Dal punto di vista pratico quello che vogliamo fare è semplicemente scambiare la prima riga con la riga contenente tale elemento. Formalmente definiamo una matrice di permutazione.

2.6.1 Matrici di Permutazione

Definizione 2.4 *Una* matrice di permutazione $\mathbf{P}$ *di ordine n é una matrice tale che:*

1. *I suoi elementi sono tutti 0 o 1.*
2. $\mathbf{P}\begin{pmatrix} 1 \\ \vdots \\ n \end{pmatrix} = \underline{v}$ *con* $\underline{v}$ *permutazione degli interi da 1..n stessi*

Proprietá 2.8 *Le matrici di permutazione hanno queste due proprietá:*

1. *Se* $\mathbf{P}$ *é una matrice di permutazione* $\Longrightarrow$ $\mathbf{P}$ *é ortogonale.*
2. *Se* $\mathbf{P}_1$ *e* $\mathbf{P}_2$ *sono matrici di permutazione* $\Longrightarrow$ *tale é anche* $\mathbf{P}_1\mathbf{P}_2$ *(la proprietá é quindi estendibile al prodotto di un numero arbitrario di matrici di permutazione).*

2.6.2 Fattorizzazione PA = LU

Tornando alla formulazione dell'algoritmo definiamo $\mathbf{P}_1$ matrice di permutazione tale che:

$$\mathbf{P}_1\begin{pmatrix} 1 \\ \vdots \\ k_1 \\ \vdots \\ n \end{pmatrix} = \begin{pmatrix} k_1 \\ \vdots \\ k_1 - 1 \\ 1 \\ k_1 + 1 \\ \vdots \\ n \end{pmatrix}$$

Allora $\mathbf{P}_1$ sará della forma:

$$\mathbf{P}_1 = \begin{bmatrix} \underline{e}_{k_1}^T \\ \vdots \\ \underline{e}_{k_1-1}^T \\ \underline{e}_1^T \\ \underline{e}_{k_1+1}^T \\ \vdots \\ \underline{e}_n^T \end{bmatrix} \longrightarrow \text{che coincide con } \mathbf{I} \text{ a meno delle righe 1 e } k_1 \text{ che sono scambiate}$$

Cioé

$$\mathbf{P}_1 = \begin{bmatrix} 0 & \dots & \dots & \dots & 1 & \dots & \dots & \dots & 0 \\ \vdots & 1 & \vdots & \vdots & 0 & \dots & \dots & \dots & \vdots \\ \vdots & \dots & \ddots & \vdots & \vdots & \dots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & 1 & \vdots & \dots & \dots & \dots & \vdots \\ 1 & 0 & \dots & \dots & 0 & \dots & \dots & \dots & \vdots \\ 0 & \dots & \dots & \dots & \dots & 1 & \vdots & \vdots & \vdots \\ \vdots & \dots & \dots & \dots & \dots & \dots & \ddots & \vdots & \vdots \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & 1 & \vdots \\ \vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 1 \end{bmatrix}$$

Dove i puntini $\vdots \dots$ sono tutti zeri. Si puó notare come $\mathbf{P}_1$ sia oltre che ortogonale (caratteristica garantita dalla prima proprietá delle matrici di permutazione) anche simmetrica, perché $(k_1,\ 1) = (1,k_1)$, per cui $\mathbf{P}_1^{-1} = \mathbf{P}_1$. Di conseguenza avremo che:

$$\mathbf{P}_1\mathbf{A}_1 = \begin{pmatrix} a^{(1)}_{k_1,1} & \dots & \dots & \dots & a^{(1)}_{k_1,n} \\ \vdots & & & & \vdots \\ a^{(1)}_{1,1} & \dots & \dots & \dots & a^{(1)}_{1,n} \\ \vdots & & & & \vdots \\ a^{(1)}_{n,1} & \dots & \dots & \dots & a^{(1)}_{n,n} \end{pmatrix}$$

In questo modo per $\mathbf{P}_1\mathbf{A}_1$ ho che $a^{(1)}_{1,1} \neq 0$ per cui

$$\underline{g}_1 = \frac{1}{a^{(1)}_{k_1,1}}\ [\ 0\ a^{(1)}_{21} \dots a^{(1)}_{k_1,1} \dots a^{(1)}_{n1}\ ]^T$$

$$\mathbf{L}_1 = \mathbf{I} - g_1\underline{e}_1^T$$

$$\mathbf{L}_1\mathbf{P}_1\mathbf{A} = \begin{pmatrix} a^{(1)}_{k_1,1} & \dots & \dots & a^{(1)}_{k_1,n} \\ 0 & a^{(2)}_{2,2} & \dots & a^{(2)}_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a^{(2)}_{n,2} & \dots & a^{(2)}_{n,n} \end{pmatrix} = \mathbf{A}^{(2)}$$

A questo punto basta iterare il procedimento come visto in precedenza con l'algoritmo di Gauss. Questo algoritmo viene detto *algoritmo di Gauss con pivoting parziale* e ci fa giungere alla forma:

$$\mathbf{L}_{n-1}\mathbf{P}_{n-1}\dots\mathbf{L}_1\mathbf{P}_1\mathbf{A} = \mathbf{A}^n = \mathbf{U}$$

con:

$$\mathbf{L}_i = \mathbf{I} - \underline{g}_i\underline{e}_i^T$$

$$\underline{g}_i = \frac{1}{a_{k_i,i}^{(i)}}\ [\ 0\dots 0 a_{k_i,i}^{(i)}\dots a_{k_n,i}^{(i)}\ ]$$

$$k_i:\quad |\ a_{k_i,i}^{(i)}\ | = \underset{k:i..n}{max}\ |\ a_{k,i}^{(i)}\ | \quad > 0$$

$$\mathbf{P}_i = \begin{bmatrix} \underline{e}_{k_i}^T \\ \vdots \\ \underline{e}_{k_i-1}^T \\ \underline{e}_i^T \\ \underline{e}_{k_i+1}^T \\ \vdots \\ \underline{e}_n^T \end{bmatrix}$$

A questo punto devo dimostrare come in precedenza che il risultato raggiunto é effettivamente una fattorizzazione LU.
Ricordando ora che $\mathbf{P}_i\mathbf{P}_i = \mathbf{P}_i^2 = \mathbf{I}$, visto che $\mathbf{P}_i = \mathbf{P}_i^{-1}$, posso scrivere il risultato a cui sono giunto

$$\mathbf{L}_{n-1}\mathbf{P}_{n-1}\dots\mathbf{L}_1\mathbf{P}_1\mathbf{A} = \mathbf{A}^n = \mathbf{U}$$

nella forma:

$\mathbf{L}_{n-1}$
$\mathbf{P}_{n-1}\mathbf{L}_{n-2}(\mathbf{P}_{n-1}$
$\mathbf{P}_{n-1})\mathbf{P}_{n-2}\mathbf{L}_{n-3}(\mathbf{P}_{n-2}\mathbf{P}_{n-1}$
$\mathbf{P}_{n-1}\mathbf{P}_{n-2})\mathbf{P}_{n-3}\mathbf{L}_{n-3}\dots$
$\vdots$
$\mathbf{P}_{n-1}\mathbf{P}_{n-2}\dots\mathbf{P}_3)\mathbf{P}_2\mathbf{L}_1(\mathbf{P}_2\dots\mathbf{P}_{n-1}$
$\mathbf{P}_{n-1}\dots\mathbf{P}_2)\mathbf{P}_1\mathbf{A}$

É importante notare come le quantità tra parentesi sono tutte uguali alla matrice identica $\mathbf{I}$. Ora definisco per ogni riga della notazione precedente una matrice $\hat{\mathbf{L}_i}$, ponendo precedentemente $\mathbf{P} = \mathbf{P}_{n-1} \dots \mathbf{P}_2\mathbf{P}_1$

$$\hat{\mathbf{L}_i} = \mathbf{P}_{n-1} \dots \mathbf{P}_{i+1}\mathbf{L}_i\mathbf{P}_{i+1} \dots \mathbf{P}_{n-1} \qquad \forall i : 1 \dots n-1$$

in questo modo posso scrivere (nota il caso i = n-1):

$$\hat{\mathbf{L}_{n-1}} \dots \hat{\mathbf{L}_1}\mathbf{PA} = \mathbf{U}$$

Come nell'algoritmo di Gauss senza pivoting adesso devo dimostrare che $\hat{\mathbf{L}_{n-1}} \dots \hat{\mathbf{L}_1} = \mathbf{L}^{-1}$, cioé che $\hat{\mathbf{L}_{n-1}} \dots \hat{\mathbf{L}_1}$ sia l'inversa di una matrice triangolare inferiore con diagonale tutta uguale a 1. A questo proposito analizziamo meglio la struttura di $\hat{\mathbf{L}_i}$

$$\hat{\mathbf{L}_i} = \mathbf{P}_{n-1} \dots \mathbf{P}_{i+1}\mathbf{L}_i\mathbf{P}_{i+1} \dots \mathbf{P}_{n-1} = \mathbf{P}_{n-1} \dots \mathbf{P}_{i+1}(\mathbf{I} - \underline{g}_i\underline{e}_i^T)(\mathbf{P}_{i+1} \dots \mathbf{P}_{n-1}) =$$

$$= (\mathbf{P}_{n-1} \dots \mathbf{P}_{i+1})(\mathbf{P}_{i+1} \dots \mathbf{P}_{n-1}) - [(\mathbf{P}_{n-1} \dots \mathbf{P}_{i+1})\underline{g}_i][\underline{e}_i^T(\mathbf{P}_{i+1} \dots \mathbf{P}_{n-1})] =$$

$$= \mathbf{I} - [(\mathbf{P}_{n-1} \dots \mathbf{P}_{i+1})\underline{g}_i][\underline{e}_i^T(\mathbf{P}_{i+1} \dots \mathbf{P}_{n-1})]$$

A questo punto chiamo $\underline{\hat{g}}_i = [(\mathbf{P}_{n-1} \dots \mathbf{P}_{i+1})\underline{g}_i]$ ed analizzo il contenuto delle seconde parentesi quadre. Intanto osservo che

$$\underline{e}_i^T\mathbf{P}_{i+1} = \underline{e}_i^T \qquad \text{e} \qquad \underline{e}_i^T\mathbf{P}_{i+2} = \underline{e}_i^T \dots\dots$$

$$\text{quindi} \qquad \underline{e}_i^T\mathbf{P}_{i+1} \dots \mathbf{P}_{n-1} = \underline{e}_i^T$$

perció in definitiva

$$\hat{\mathbf{L}_i} = \mathbf{I} - \underline{\hat{g}}_i\underline{e}_i^T$$

mi rimane da analizzare il significato di $\underline{\hat{g}}_i$ ed in particolare dobbiamo dimostrare che $\hat{\mathbf{L}_i}$ e $\mathbf{L}_i$ hanno la stessa struttura di zeri. A questo scopo dimostro che $\underline{\hat{g}}_i$ e $\underline{g}_i$ hanno la stessa struttura di zeri (cioé entrambi hanno tutti zeri nelle prime i posizioni), infatti:

$$\mathbf{P}_{i+1}g_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ * \\ \vdots \\ * \end{pmatrix}$$

Infatti la matrice $\mathbf{P}_{i+1}$ permuta soltanto gli elementi del vettore g_i con indice da i+1 a n, lasciando quindi inalterati i primi i 0 del vettore stesso. A maggior ragione questo avverrá per $\mathbf{P}_{i+2}\ldots$ per cui in generale $\underline{\hat{g}}_i = [(\mathbf{P}_{n-1}\ldots\mathbf{P}_{i+1})\underline{g}_i]$ avrá nelle sue prime i posizioni tutti zeri, cioé ha la stessa struttura di zeri di $\underline{g}_i$, come volevamo appunto dimostrare. Da questo deriva che possiamo affermare dal punto di vista formale che $\hat{\mathbf{L}}_i$ e $\mathbf{L}_i$ hanno la stessa struttura per cui possiamo scrivere

$$\mathbf{L}^{-1}\mathbf{PA} = \mathbf{U} \Longrightarrow \mathbf{PA} = \mathbf{LU}$$

perció possiamo scrivere:

$$\mathbf{A}x = b \Longrightarrow \mathbf{PA}x = \mathbf{P}b \Longrightarrow \mathbf{LU}x = \mathbf{P}b$$

dove il termine noto é permutato rispetto a b.

2.6.3 Costi dell'algoritmo

Il costo di tale fattorizzazione rimane $\approx \frac{2}{3}n^3$ flops, mentre la permutazione $\mathbf{P}$ ha evidentemente un costo proporzionale a n (quindi trascurabile rispetto a n^3). Rimane quindi da analizzare il costo per la costruzione di $\mathbf{P}$. Ricordiamo che $\mathbf{P} = \mathbf{P}_{n-1}\ldots\mathbf{P}_1$ con:

$$\mathbf{P}_i \begin{pmatrix} 1 \\ \vdots \\ n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ k_i \\ \vdots \\ i \\ \vdots \\ n \end{pmatrix} \longrightarrow \text{con i in riga } k_i \text{ e } k_i \text{ in riga i e}$$

$$k_i : \quad | \, a^{(i)}_{k_i,i} \, | = \max_{k:i..n} | \, a^{(i)}_{k_i} \, |$$

perció per ogni k_i ho $n-i$ confronti per cui globalmente

$$\sum_{i=1}^{n-1}(n-i) \approx \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \approx \frac{n^2}{2}$$

che é ancora trascurabile rispetto $\frac{2}{3}n^3$.
In definitiva quindi il costo dell'algoritmo di Gauss con pivoting parziale é comunque di $\approx \frac{2}{3}n^3$ flops.
Per quanto riguarda il costo dell'algoritmo in termini di occupazione di memoria osserviamo che:

$$P \begin{pmatrix} 1 \\ \vdots \\ n \end{pmatrix} = \begin{pmatrix} i_1 \\ \vdots \\ i_n \end{pmatrix} = \underline{v}$$

quindi mi basta tenere traccia del vettore $\underline{v}$ di n elementi per memorizzare tutta l'informazione della matrice di permutazione P.

2.7 Matrici a diagonale dominante e fattor. LDL

Esistono due classi di matrici per cui la non singolaritá implica la non singolaritá di ogni sottomatrice principale, per le quali cioé

$$\det(\mathbf{A}) \neq 0 \Longrightarrow \det(\mathbf{A_i}) \neq 0 \quad \forall i : 1 \ldots n-1$$

Tali classi di matrici sono:

1. Matrici a diagonale dominante
2. Matrici simmetriche definite positive

2.7.1 Matrici a diagonale dominante

Definizione 2.5 *Sia $A = (a_{ij}) \in \Re^{(nxn)}$ é a diagonale dominante sse*

$$\forall i : 1 \ldots n \qquad | a_{ii} | > \sum_{j=1, j \neq i}^{n} | a_{ij} |$$

Teorema 2.9 *Se $\mathbf{A}$ é diagonale dominante $\Longrightarrow \forall k = 1 \ldots n \quad \mathbf{A}_k$ é diagonale dominante.*

Teorema 2.10 *Se $\mathbf{A}$ é diagonale dominante $\Longrightarrow \mathbf{A}$ é non singolare.*

La piú immediata conseguenza dei due precedenti teoremi é che se una matrice $\mathbf{A}$ é a diagonale dominante allora é fattorizzabile LU. Infatti se $\mathbf{A}$ é a diagonale dominante $\Longrightarrow$ é non singolare per il secondo teorema e, per il primo teorema, anche tutte le sue sottomatrici principali sono non singolari: questa é la condizione sufficiente perché la matrice sia fattorizzabile LU e quindi vi si puó applicare l'algoritmo di Gauss senza pivoting. In simboli:

$$\mathbf{A} \text{ diagonale dominante } \Longrightarrow (det(\mathbf{A}) \neq 0) \wedge (\forall k = 1 \ldots n \quad det(\mathbf{A}_k) \neq 0)$$

$$\Longrightarrow \mathbf{A} \text{ fattorizzabile LU}$$

La dimostrazione del primo teorema é banale e lasciata per esercizio. Ecco invece la dimostrazione del secondo.

Dimostrazione: Supponiamo, per assurdo, che la matrice $\mathbf{A}$ diagonale dominante sia singolare, per cui $det(\mathbf{A}) = 0$ allora

$$\exists \underline{x} \in \Re^n, \quad \underline{x} \neq \underline{O} : \quad \mathbf{A}\underline{x} = \underline{O}$$

Sia $i :| x_i | = \max_{j:1..n} | x_j | > 0$,sicuramente positivo visto che $\underline{x} \neq \underline{O}$. A questo punto posso scrivere che

$$\underline{e}_i^T \mathbf{A}\underline{x} = \underline{e}_i^T \underline{O} = \underline{O}$$

Osservo che la scrittura $\underline{e}_i^T\mathbf{A}$ non rappresenta altro che la i-esima riga della matrice $\mathbf{A}$ per cui posso scrivere

$$\sum_{j=1}^{n} a_{ij}x_j = 0 \Longrightarrow a_{ii}x_i = -\sum_{j=1,j\neq i}^{n} a_{ij}x_j$$

dove ho separato l'elemento a_{ii} dalla sommatoria, per cui dividendo entrambi i membri per x_i

$$a_{ii} = -\sum_{j=1,j\neq i}^{n} a_{ij}\frac{x_j}{x_i} \qquad \longrightarrow x_i \neq 0 \text{ sicuramente}$$

$$\Longrightarrow \mid a_{ii} \mid = \mid \sum_{j=1,j\neq i}^{n} a_{ij}\frac{x_j}{x_i} \mid \leq \sum_{j=1,j\neq i}^{n} \mid a_{ij}\frac{x_j}{x_i} \mid = \sum_{j=1,j\neq i}^{n} \mid a_{ij} \mid\mid \frac{x_j}{x_i} \mid$$

Per come si é definito x_i abbiamo che $\mid \frac{x_j}{x_i} \mid \leq 1$ allora

$$\mid a_{ii} \mid \leq \sum_{j=1,j\neq i}^{n} \mid a_{ij} \mid$$

e questo ci porta all'assurdo visto che ci porterebbe a dire che la matrice $\mathbf{A}$ non é a diagonale dominante, contrariamente alle ipotesi. Siccome l'assurdo sta nell'aver posto $det(\mathbf{A}) = 0$ il teorema é dimostrato.

2.7.2 Matrici singolari e definite positive

Definizione 2.6 *Una matrice* $\mathbf{A}$ *é simmetrica definita positiva sse* $\mathbf{A}$ *é simmetrica, cioé* $\mathbf{A} = \mathbf{A}^T$ *,e* $\forall \underline{x} \in \Re^n, \underline{x} \neq \underline{O} : \quad \underline{x}^T\mathbf{A}\underline{x} > 0.$

Da notare che $\underline{x}^T\mathbf{A}\underline{x}$ é uno scalare perché le dimensioni dei fattori sono rispettivamente $(1xn), (nxn), (nx1)$. A proposito delle matrici simmetriche definite positive abbiamo tre teoremi:

Teorema 2.11 *Se* $\mathbf{A}$ *é simmetrica definita positiva* $\Longrightarrow \forall k : 1\ldots n : \mathbf{A}_k$ *é simmetrica definita positiva.*

Teorema 2.12 *Se* $\mathbf{A}$ *é simmetrica definita positiva* $\Longrightarrow det(\mathbf{A}) \neq 0$*, cioé* $\mathbf{A}$ *é non singolare.*

Teorema 2.13 *Se* $\mathbf{A}$ *é simmetrica definita positiva* $\Longrightarrow a_{ii} > 0 \quad \forall i : 1\ldots n.$

Le dimostrazioni dei tre teoremi vengono date in ordine inverso. Ecco la dimostrazione dell'ultimo
Dimostrazione: Semplicemente:

$$a_{ii} = \underline{e}_i^T\mathbf{A}\underline{e}_i > 0$$

Ecco la dimostrazione del secondo teorema

Dimostrazione: Supponiamo, per assurdo, che la matrice $\mathbf{A}$, simmetrica definita positiva, sia singolare. Allora

$$det(\mathbf{A}) = 0 \Longrightarrow \exists \underline{x} \neq 0 : \mathbf{A}\underline{x} = \underline{O}$$

A questo punto sfrutto il fatto che la matrice sia simmetrica definita positiva

$$\underline{x}^T \mathbf{A} \underline{x} = \underline{x}^T \underline{O} = 0$$

ma siccome $\underline{x} \neq 0$ per ipostesi, la matrice non puó essere simmetrica definita positiva, per cui si arriva all'assurdo. Visto che l'assurdo risiede nell'aver ipotizzato la singolaritá della matrice il teorema é dimostrato.

E finalmente la dimostrazione del terzo teorema

Dimostrazione: Se prendo una matrice $\mathbf{A}$ simmetrica definita positiva é abbastanza semplice dimostrare che tutte le sue sottomatrici principali sono tutte simmetriche, infatti é evidente che se $\mathbf{A}$ é simmetrica anche le $\mathbf{A}_k \forall k : 1 \dots n$ siano simmetriche, cioé $\mathbf{A}_k^T = \mathbf{A}_k$. Piú complessa é la dimstrazione che le sottomatrici sono anche definite positive. Rimane quindi da provare che

$$\forall \underline{\tilde{x}} \in \Re^k, \quad \underline{\tilde{x}} \neq 0 : \quad \underline{\tilde{x}}^T \mathbf{A} \underline{\tilde{x}} > 0$$

Se, per assurdo, non fosse cosí allora

$$\exists \underline{\bar{x}} \in \Re^k : \quad \underline{\bar{x}}^T \mathbf{A} \underline{\bar{x}} \quad \text{con } \underline{\bar{x}} \neq 0$$

definisco allora un vettore $\underline{x}$ di n elementi in questo modo

$$\underline{x} = \begin{pmatrix} \underline{\bar{x}} \\ \\ \underline{O}_{n-k} \end{pmatrix} \neq \underline{O} \in \Re^n$$

Sfruttando la simmetria posso scrivere la matrice $\mathbf{A}$ come

$$\mathbf{A} = \begin{bmatrix} A_k & B^T \\ \\ B & D \end{bmatrix}$$

Tale che $B \in \Re^{(n-k)x(k)}$ e $D \in \Re^{(n-k)x(n-k)}$.
per cui ho

$$0 < \underline{x}^T \mathbf{A} \underline{x} = (\underline{\bar{x}}^T \underline{O}^T) \begin{bmatrix} A_k & B^T \\ \\ B & D \end{bmatrix} \begin{pmatrix} \underline{\bar{x}} \\ \\ \underline{O}_{n-k} \end{pmatrix} = (\underline{\bar{x}}^T \underline{O}^T) \begin{pmatrix} \mathbf{A}_k \underline{\bar{x}} \\ \\ \mathbf{B} \underline{\bar{x}} \end{pmatrix} = \underline{\bar{x}}^T \mathbf{A} \underline{\bar{x}}$$

Ma quest'ultima quantitá per le ipotesi iniziali doveva essere ≤ 0 (visto che per ipotesi la matrice $\mathbf{A}$ doveva essere simmetrica definita positiva) ed invece si trova ad essere > 0. Questo ci porta ad un assurdo e quindi alla dimostrazione del teorema.

2.7.3 Fattorizzazione LDL

Le proprietá delle matrici simmetriche definite positive, come nel caso di quelle a diagonale dominante, garantiscono che tali matrici siano fattorizzabili LU, quindi posso scrivere $\mathbf{A}$ sdp $\Longrightarrow \mathbf{A} = \mathbf{LU}$.
Per quanto riguarda le matrici simmetriche definite positive peró possiamo fare qualcosa di piú sfruttando il fatto che essendo esse simmetriche é sufficiente conoscerne una porzione triangolare (superiore o inferiore).
Quindi ho che $\mathbf{A} = \mathbf{LU}$ e $\mathbf{L}$ triangolare inferiore con diagonale uguale a 1 e

$$\mathbf{U} = \begin{pmatrix} u_{11} & \dots & \dots & u_{1n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & u_{nn} \end{pmatrix} = \begin{pmatrix} u_{11} & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & u_{nn} \end{pmatrix} \begin{pmatrix} 1 & u_{12} & \dots & u_{nn} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \dots & 0 & 1 \end{pmatrix} \equiv \mathbf{D}\hat{\mathbf{U}}$$

Adesso sfrutto il fatto che $\mathbf{A}$ sia simmetrica e $\mathbf{D}$ sia diagonale (per cui $\mathbf{D} = \mathbf{D}^T$) per scrivere

$$\mathbf{A} = \mathbf{LD}\hat{\mathbf{U}} = \mathbf{A}^T = \hat{\mathbf{U}}^T\mathbf{DL}^T$$

In definitiva quindi

$$\mathbf{A} = \mathbf{LD}\hat{\mathbf{U}} = \hat{\mathbf{U}}^T\mathbf{DL}^T$$

A questo punto siccome sia $\mathbf{L}$ che $\hat{\mathbf{U}}^T$ sono triangolari inferiori con diagonale uguale ad 1 ed essendo $\mathbf{A}$ fattorizzabile LU ed essendo tale fattorizzazione unica per un teorema precedentemente dimostrato posso affermare che

$$\mathbf{L} = \hat{\mathbf{U}}^T \quad \text{oppure} \quad \mathbf{L}^T = \hat{\mathbf{U}}$$

per cui sono giunto alla formula

$$\mathbf{A} = \mathbf{LU} = \mathbf{LDL}^T$$

che viene chiamata appunto *Fattorizzazione LDL.*
Da questa formula si vede anche come in realtá per memorizzare $\mathbf{A}$ mi sia sufficiente memorizzare la sua porzione diagonale inferiore ponendoci gli elementi sottodiagonali di $\mathbf{L}$ e gli elementi diagonali di $\mathbf{D}$. Inoltre si puó dimostrare che il numero di flops necessari per trovare tale fattorizzazione passa da un ordine di $\frac{2}{3}n^3$ a $\frac{1}{3}n^3$.

Corollario 2.14 *Se* $\mathbf{A}$ *é simmetrica definita positiva allora* $\mathbf{D}$ *é anch'essa simmetrica definita positiva, per cui per il terzo teorema sulle matrici s.d.p.*

$$\mathbf{D} = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix} : \quad d_i > 0 \quad \forall i : 1 \dots n$$

Dimostrazione: Se $\mathbf{A}$ é s.d.p. allora é fattorizzabile LDL per cui

$$\mathbf{A} = \mathbf{LDL}^T \Longrightarrow \mathbf{D} = \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}$$

Con $\mathbf{L}^{-T} = (\mathbf{L}^{-1})^T = (\mathbf{L}^T)^{-1}$. Per dimostrare il corollario devo verificare che

$$\forall \underline{x} \neq 0 : \quad \underline{x}^T\mathbf{D}\underline{x} > 0$$

Procedendo per sostituzioni successive

$$\Longrightarrow \underline{x}^T\mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}\underline{x} > 0 \Longrightarrow (\underline{x}^T\mathbf{L}^{-1})\mathbf{A}(\mathbf{L}^{-T}\underline{x}) > 0$$

a questo punto pongo $\underline{y} = \mathbf{L}^{-T}\underline{x}$ ed ottengo l'espressione

$$\underline{y}^T\mathbf{A}\underline{y} > 0$$

che é vera sicuramente, per il fatto che $\mathbf{A}$ é s.d.p., se dimostro che $\underline{y} \neq 0$. Infatti

$$y = \mathbf{L}^{-T}\underline{x} \Longrightarrow \mathbf{L}^T\underline{y} = \underline{x}$$

e questo significa proprio che $\underline{y} \neq 0$, perché altrimenti avrei che $\mathbf{L}$ e $\mathbf{A}$ sarebbero singolari.

2.7.4 Algortimo di fattorizzazione LDL

Per quanto visto in precedenza abbiamo che se una matrice é simmetrica definita positiva allora é fattorizzabile LU e in particolare é fattorizzabile LDL. In simboli

$$\mathbf{A}\ s.d.p. \Longrightarrow \mathbf{A} = \mathbf{LU} \Longrightarrow \mathbf{A} = \mathbf{LDL}^T$$

$$con\ D = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix} : \quad d_i > 0 \quad \forall i : 1 \dots n$$

Quello che vorrei é ottenere un metodo che partendo dalla matrice di partenza mi porti a determinare le matrici $\mathbf{L}$ e $\mathbf{D}$. Siccome so che $\mathbf{A} = \mathbf{LDL}^T$ quello che mi interessa é la parte inferiore della matrice, per cui posso supporre $\underline{j \leq i}$ e considero le seguenti trasformazioni

$$a_{ij} = \underline{e}_i^T\mathbf{A}\underline{e}_j = \underline{e}_i^T(\mathbf{LDL}^T)\underline{e}_j = (\underline{e}_i^T\mathbf{L})\mathbf{D}(\underline{e}_j^T\mathbf{L})^T =$$

$$= (l_{i1}l_{i2}\dots l_{ii}0\dots 0)\mathbf{D}(l_{j1}l_{j2}\dots l_{jj}0\dots 0)^T =$$

$$= (l_{i1}l_{i2}\dots d_i 0\dots 0)\begin{pmatrix} l_{j1} \\ \vdots \\ l_{jj} \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \sum_{k=1}^{min(i,j)} l_{ik}d_kl_{jk}$$

Ma siccome ho supposto $j \leq i$ allora

$$a_{ij} = \sum_{k=1}^{j} l_{ik} d_k l_{jk}$$

A questo punto devo considerare i due casi $i = j$ e $i \leq j$
1) Se $i = j$ allora

$$a_{jj} = \sum_{k=1}^{j} (l_{jk})^2 d_k = \sum_{k=1}^{j-1} (l_{jk})^2 d_k - (l_{jj})^2 d_j$$

Siccome $l_{jj} = 1$ allora

$$a_{jj} = \sum_{k=1}^{j-1} (l_{jk})^2 d_k - d_j$$

per cui

$$d_j = a_{jj} - \sum_{k=1}^{j-1} (l_{jk})^2 d_k$$

2) Se $j < i$

$$a_{ij} = \sum_{k=1}^{j-1} l_{ik} d_k l_{jk} + l_{ij} d_j l_{jj}$$

Siccome $l_{jj} = 1$ allora

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_k l_{jk}}{dj}$$

Consideriamo ora alcuni casi esemplificativi. Se $i = j = 1$ le sommatorie non contribuiscono, quindi $d_i = a_{ii}$, e svolgendo i calcoli lungo la prima colonna, si ha che $l_{i1} = \frac{a_{i1}}{d_1}$.
Proseguendo con $j = 2$ ho che $d_2 = a_{22} - (l_{21})^2 d_1$. É importante notare che $(l_{21})^2$ e d_1 sono giá stati calcolati in precedenza per cui sono giá disponibili. Di seguito ho che per $i > 2$

$$l_{i2} = \frac{a_{i2} - l_{i1} d_1 l_{21}}{d_2}$$

ed anche qui é importante notare che tutti i valori che ci servono sono giá stati calcolati e sono pertanto giá disponibili.

2.7.5 Costo dell'algortimo di fattorizzazione LDL

Per quanto riguarda il costo dell'algoritmo di fattorizzazione LDL considero soltanto il caso in cui $j < i$, perché nel caso $i = j$ si ha una complessitá dell'ordine di n, per cui trascurabile.
Nel caso appunto in cui $j < i$ ho la formula

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_k l_{jk}}{dj}$$

posso peró notare che sia la quantitá $\eta_{jk} = d_k l_{jk}$ che d_j sono giá stati calcolati in passi precedenti e quindi puó stare in memoria temporanea e possiamo assumere il costo del loro calcolo pari a 0. Ci rimane quindi un prodotto $(l_{ik}\eta_{jk})$ ad ogni somma della sommatoria, per cui due flops per somma. Siccome si eseguono $j-1$ somme ho che il costo del calcolo di l_{ij} é $2+2(j-1) = 2j$. In totale ho che questo calcolo va eseguito $n-j$ volte per colonna della matrice, per cui ho $2j(n-j)$ operazioni, per cui il costo totale sará

$$\sum_{j=1}^{n}(2j)(n-j) = 2n\sum_{j=1}^{n} j - 2\sum_{j=1}^{n} j^2 = \frac{2n(n)(n+1)}{2} - \frac{2n(n+1)(2n+1)}{6} \approx$$

$$\approx n^3 - \frac{2}{3}n^3 = \frac{1}{3}n^3$$

Quindi come si vede il costo di questo algoritmo é come ordine piú basso rispetto all'algoritmo di fattorizzazione LU che aveva un costo dell'ordine di $\frac{2}{3}n^3$.

Capitolo 3

Metodi di risoluzione di equazioni non lineari

3.1 Introduzione

Il problema che ci poniamo in questo capitolo è il seguente: data una funzione

$$f : I \subseteq \mathbf{R} \to \mathbf{R}$$

cerchiamo le sue *radici*, ossia quei punti $\overline{x} \in \mathbf{R}$ tali che $f(\overline{x}) = 0$.

In questo caso, diversamente dai sistemi lineari, non abbiamo un criterio per decidere l'esistenza e l'unicità. Infatti si possono avere situazioni molto variegate: ad esempio le funzioni $f(x) = x^2 + 1$, $g(x) = x^2 - 1$, $h(x) = \sin(x)$ hanno rispettivamente 0, 2 e infinite soluzioni reali.

Quindi ci accontenteremo di determinare solo una di queste radici (se esiste).

3.2 Metodo di bisezione

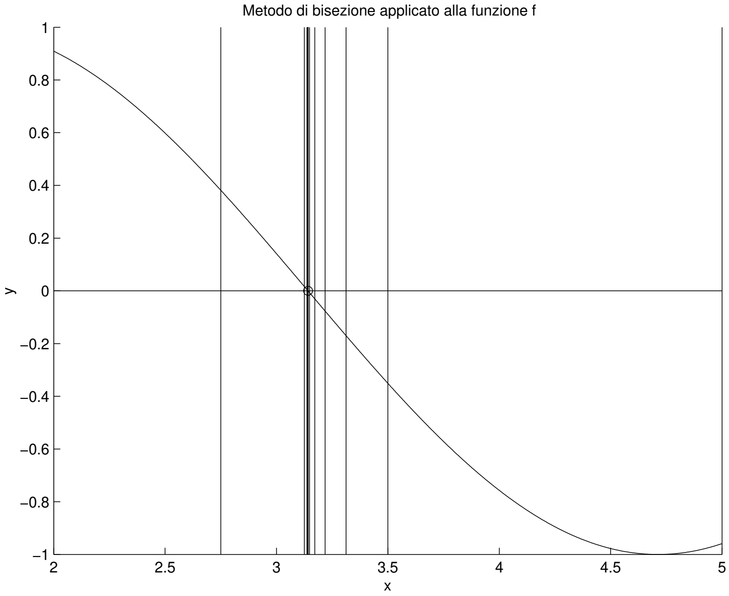
Supponiamo che f sia una funzione continua nell'intervallo $[a, b]$ tale che $f(a)f(b) < 0$ cioè la funzione assume valori di segno opposto nei due estremi. Allora per il *teorema dell'esistenza degli zeri* $\exists c : a \leq c \leq b, f(c) = 0$. Naturalmente nell'interrvallo può esserci più di una radice: il teorema ci assicura che ce n'è almeno una.

Definiamo $x_1 = \frac{a+b}{2}$ allora si può avere uno dei seguenti tre casi:

1. $f(x_1) = 0$ allora abbiamo avuto fortuna: x_1 è una radice e l'algoritmo termina.

2. $f(a)f(x_1) < 0$ allora possiamo reiterare il procedimento sul nuovo intervallo $[a, x_1]$.

3. $f(b)f(x_1), 0$ allora possiamo reiterare il procedimento sul nuovo intervallo $[x_1, b]$.



In un passo abbiamo in questo modo dimezzato l'ampiezza dell'intervallo di confidenza, quindi reiterando questo metodo è possibile trovare un intervallo di confidenza di ampiezza piccola a piacere.

Formalmente il metodo può essere descritto in questo modo:

- $[a_1, b_1] = [a, b]$ tale che $f(a)f(b) < 0$.
- per $i = 1, 2, \ldots$ eseguire i seguenti passi
  - $x_i = \frac{a_i+bi}{2}$
  - Valutazione di $f(x_i)$
  - Se $f(x_i) = 0$ allora ESCI
    altrimenti se $f(a_i)f(x_i) < 0$
    - $[a_{i+1}, b_{i+1}] = [a_i, x_i]$

    altrimenti
    - $[a_{i+1}, b_{i+1}] = [x_i, b]$

    fine se

  fine per

Supponiamo di essere interessati ad una approssimazione x_i tale che

$$|\overline{x} - x_i| < \epsilon \tag{3.1}$$

dove $\overline{x}$ è il valore esatto della radice cercata.

Poiché $x_1 = \frac{a+b}{2}$ si ha $|\overline{x} - x_1| < \frac{b-a}{2}$. In generale $|\overline{x} - x_i| < \frac{b_i - a_i}{2}$. Sviluppando questa relazione si ottiene

$$\begin{aligned} |\overline{x} - x_i| < \frac{b-a}{2^i} &\leq \epsilon \\ b - a &\leq \epsilon 2^i \\ \frac{b-a}{\epsilon} &\leq 2^i \\ \log_2 b - a - \log_2(\epsilon) &\leq i \\ i \geq \lceil \log_2(b-a) - \log_2(\epsilon) \rceil &= \nu \end{aligned} \tag{3.2}$$

In questo modo, fissata la accuratezza ϵ, viene fissato anche il numero massimo di iterazioni, ν. Questa proprietà è molto importante, perchè ci permette di essere sicuri della terminazione dell'algoritmo dopo ν passi. Il numero di passi non deve essere deciso a priori dall'esterno ma può essere calcolato in funzione della accuratezza ϵ. Questo risultato ci permette di sostituire nella descrizione la riga "per $i = 1, 2, \ldots$" con "per $i = 1..\nu$". In questo modo l'algoritmo termnerà sicuramente dopo ν passi, ma terminerà prima nel caso che si trovi una radice in un numero minore di passi.

Un altro problema che si presenta applicando alla lettera la prima approssimazione dell'algoritmo che abbiamo scritto è la condizione $f(x_i) = 0$ che è difficilmente verificabile in aritmetica finita e dev'essere più realisticamente sostituita con una condizione del tipo $|f(x_i)| < \delta$. Il problema che ci proponiamo di risolvere ora è quello di capire quanto deve essere piccolo δ per ottenere la accuratezza ϵ desiderata. Più formalmente vogliamo determinare

$$\delta = \delta(\epsilon) \text{ tale che } |f(x)| < \delta \Longrightarrow |\overline{x} - x_i| < \epsilon \tag{3.3}$$

Per ricavare questo δ sviluppiamo secondo la formula di Taylor la f nel punto $\overline{x}$ (cioè nel punto tale che $f(\overline{x}) = 0$), fermandoci ad un'approssimazione del primo ordine.

$$f(x_i) \approx f(\overline{x}) + f'(\overline{x})(\overline{x} - x_i) \tag{3.4}$$

Ma $f(\overline{x}) = 0$ quindi otteniamo

$$\begin{aligned} f(x_i) &\approx f'(\overline{x})(\overline{x} - x_i) \\ (\overline{x} - xi) &\approx \frac{f(x_i)}{f'(\overline{x})} \end{aligned} \tag{3.5}$$

Passando ai valori assoluti

$$|\overline{x} - x_i| \approx \left| \frac{f(x_i)}{f'(\overline{x})} \right| \tag{3.6}$$

Questa relazione richiede di conoscere la derivata di f in $\overline{x}$, ma $\overline{x}$ è incognita (è proprio quello che stiamo cercando!) quindi dobbiamo cercare una sua approssimazione: l'idea più ovvia è approssimare $f'(\overline{x})$ con $f'(x_i)$. Otteniamo quindi, ricordando che $|f(x_i)| < \delta$,

$$|x - x_i| \approx \frac{|f(x_i)|}{|f'(x_i)|} \leq \frac{\delta}{|f'(\overline{x})|} \leq \epsilon \tag{3.7}$$

Dalla (3.7) si ottiene

$$\delta \approx \epsilon |f'(x_i)| \tag{3.8}$$

Tuttavia calcolare esplicitamente $f'(x_i)$ sarebbe più costoso del metodo di bisezione stesso. Quello che possiamo fare è approssimare $f'(x_i)$ con il rapporto incrementale che ha come estremi l'intervallo corrente $[a_i, b_i]$:

$$\delta \approx \epsilon \left| \frac{f(b_i) - f(a_i)}{b_i - a_i} \right| \tag{3.9}$$

Il calcolo di questo rapporto incrementale non comporta molto costo aggiuntivo perche i valori di $f(a)$ e $f(b)$ sono noti. Dal punto di vista implementativo si nota che ad ogni passo dell'algoritmo è necessario valutare la funzione solo in un punto perchè un estremo dell'intervallo considerato non cambia.

3.3 Condizionamento del problema

Vediamo ora qual è l'interpretazione geometrica della relazione

$$|\overline{x} - x_i| \lessapprox \frac{\delta}{|f'(\overline{x})|} \tag{3.10}$$

Nella (3.10) δ può essere vista come l'errore indotto sulla variabile dipendente, quello che in altro contesto avevamo chiamato ϵ_y (mentre ϵ corrisponde a ϵ_x). Allora possiamo considerare $1/f'(\overline{x})$ il *numero di condizionamento del problema* Il numero di condizionamento della radice allora è

$$k = \frac{1}{|f'(\overline{x})|} \tag{3.11}$$

Il problema è ben condizionato cioè k è piccolo se si ha $|f'(\overline{x})| \gg 0$ Dal punto di vista geometrico questo corrisponde ad una funzione la cui tangente nella radice è molto inclinata e quindi un intervallo grande sulla y (un δ grande) diventa piccolo sulla x (ϵ piccolo).

Il problema è mal condizionato se k è grande cioè se $f'(\overline{x}) \approx 0$. In questo caso la tangente è quasi parallela all'asse x quindi un intervallo piccolo sull'asse y corrisponde ad un intervallo di confidenza molto grande sulle x.

Da queste considerazioni si vede subito che la ricerca di radici multiple è un problema mal condizionato. Infatti sappiamo (forse sarebbe meglio dire *dovremmo sapere*?) dall'analisi che

Teorema 3.1 *Se $\overline{x}$ è una radice di molteplicita m per $f(x)$ allora si ha*

$$f'(\overline{x}) = 0, \ldots, f^{(m-1)}(\overline{x}) = 0 \tag{3.12}$$

Se $m = 1$, cioè la radice è semplice, la (3.12) si riduce a $f(\overline{x}) = 0$; se $m > 1$ si ha $f'(\overline{x}) = 0 \Longrightarrow k \to \infty$.

Se sviluppiamo in serie di Taylor $f(x)$ nel punto $\overline{x}$, si ottiene

$$f(x) = \sum_{n \in \mathbf{N}} \frac{f^{(n)}(\overline{x})}{n!}(x - \overline{x})^n \tag{3.13}$$

Se $\overline{x}$ è una radice di molteplicità m i primi m termini della serie sono nulli (proprietà 3.12) quindi si può scrivere

$$f(x) = \sum_{n=m}^{\infty} \frac{f^{(n)}(\overline{x})}{n!}(x - \overline{x})^n = (x - \overline{x})^m \sum_{n=0}^{\infty} \frac{f^{(n)}(\overline{x})}{n!}(x - \overline{x})^n \tag{3.14}$$

cioè in prossimità della radice

$$f(x) \approx \alpha(x - \overline{x})^m \tag{3.15}$$

la f è assimilabile ad un polinomio di grado m quindi maggiore è m, più la funzione è appiattita sull'asse x.

3.4 Ordine di convergenza di un metodo iterativo

Il numero di iterazioni che un metodo iterativo richiede per ottenere una certa accuratezza dà una misura dell'efficienza del metodo. Diamo però una definizione più formale che ci permetta di stabilire se un metodo converge più o meno velocemente di un altro.

Abbiamo visto che un metodo iterativo può essere visto come una funzione Φ che lega l'approssimazione al passo $(i + 1)$-esimo con quella al passo i-esimo.

$$x_{i+1} = \Phi(x_i) \tag{3.16}$$

Un metodo iterativo si dice *convergente* se $x_i \to \overline{x}$ per $i \to \infty$ dove $\overline{x}$ è la soluzione esatta (ad esempio il valore della radice cercata di $f(x)$).

Definiamo l'*errore al passo i-esimo*

$$e_i = \overline{x} - x_i \tag{3.17}$$

Notiamo che la convergenza implica che $\lim_{i \to \infty} e_i = 0$.

Il metodo iterativo si dice *convergente di ordine p* se

$$\lim_{i \to \infty} \frac{e_{i+1}}{e_i^p} = c \tag{3.18}$$

dove c è una costante diversa da 0 ($c \neq 0, |c| < \infty$)

Nel caso che $p = 1$ dev'essere anche $|c| < 1$ altrimenti non c'è convergenza. Se vale la convergenza di ordine p,

$$e_i \to 0 \frac{e_{i+1}}{e_i^p} \to c \tag{3.19}$$

allora si ha

$$i \gg 0 \Longrightarrow e_i \approx 0, e_{i+1} \approx c e_i^p \tag{3.20}$$

Esempio: Supponiamo $c = 0.1$ e confrontiamo un metodo con convergenza lineare ($p = 1$) e uno con convergenza quadratica ($p = 2$) partendo dallo stesso errore iniziale 0.1.

	$p = 1$	$p = 2$
e_1	0.1	0.1
e_2	0.01	0.001
e_3	10^{-3}	10^{-7}
e_4	10^{-4}	10^{-15}

Dalla tabella si vede che un ordine di convergenza più elevato fa diminuire molto più velocemente l'errore.

3.5 Ordine di convergenza del metodo di bisezione

Mostriamo che il metodo di bisezione ha un ordine di convergenza lineare. Per l'errore al passo i-esimo vale la seguente relazione

$$e_i = \frac{b_i - a_i}{2} \tag{3.21}$$

analogamente, al passo $(i + 1)$-esimo:

$$e_{i+1} = \frac{b_{i+1} - a_{i+1}}{2} = \frac{b_i - a_i}{2^2} \tag{3.22}$$

Quindi

$$\lim_{i \to \infty} \frac{e_{i+1}}{e_i} = \frac{b_i - a_i}{4} \frac{2}{b_i - a_i} = \frac{1}{2} \tag{3.23}$$

■

3.6 Applicazioni del metodo di Newton

Supponiamo di voler calcolare $\overline{x} = \sqrt{a}$ con $a > 0$,
in particolare ricerchiamo la radice positiva.
$x = \sqrt{a} \leftrightarrow x^2 = a \leftrightarrow x^2 - a = 0$
per cui il problema diventa trovare la radice di $f(x) = x^2 - a = 0$.
A tale scopo ci viene incontro il metodo di Newton:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2}\frac{2x_n^2 - x_n^2 + a}{x_n} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

Si ha quindi il seguente schema iterativo :

$$\begin{cases} x_0 = \alpha & \text{qualunque} \\ x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \end{cases}$$

Si puó dimostrare che per questo particolare problema la convergenza é globale.In particolare la successione converge quadraticamente alla soluzione, poiché le radici sono sempre semplici.

Esercizio:
Generalizzare la costruzione della macchina iterativa per il problema $\sqrt[n]{a}$

3.7 Modifica del metodo di Newton per radici multiple

Abbiamo visto che il metodo di Newton converge linearmente a radici multiple. Se $x_n \to \overline{x_i}$ con $\overline{x_i}$ di molteplicitá r, allora la seguente modifica ripristina la convergenza quadratica del metodo di Newton:

$$x_{n+1} = x_n - r\frac{f(x_n)}{f'(x_n)}$$

Infatti abbiamo visto che se $\overline{x}$ ha molteplicitá r, allora per $x \approx \overline{x}$ si ha che $f(x) \approx \alpha\,(x - \overline{x})^r$ (sviluppo di Taylor). Applichiamo il metodo di Newton a questa funzione approssimata:

$$x_{n+1} = x_n - r\frac{\alpha\,(x_n - \overline{x})^r}{r\alpha\,(x_n - \overline{x})^{r-1}} = x_n - (x_n - \overline{x}) = \overline{x}$$

Quindi se r é noto non ci sono problemi a ripristinare la convergenza quadratica.

3.8 Metodo di accelerazione di Artken

Ricordiamo che la successione converge linearmente se

$$\lim_{n\to\infty} \frac{e_{n+1}}{e_n} = c \neq 0 \text{ e } |c| < \infty \text{ con } e_n = x_n - \overline{x}$$

Supponiamo di avere generato x_{n-1}, x_n, x_{n+1} per cui:

$$e_{n+1} \approx ce_n$$

$$e_n \approx ce_{n-1}$$

dove c é incognita.
Per eliminare c dividiamo membro a membro:

$$\frac{e_{n+1}}{e_n} \approx \frac{e_n}{e_{n-1}}$$

ma gli e_n sono in funzione di $\overline{x}$ che é incognita. Scrivendo per esteso:

$$\frac{x_{n+1} - \overline{x}}{x_n - \overline{x}} \approx \frac{x_n - \overline{x}}{x_{n-1} - \overline{x}}$$

Abbiamo ottenuto un'equazione nell'incognita $\overline{x}$. Calcoliamo $\overline{x}$ e la prendiamo come nuova approssimazione:

$$(x_{n+1} - \overline{x})(x_{n-1} - \overline{x}) \approx (x_n - \overline{x})^2$$

Svolgendo i calcoli si ottiene:

$$\overline{x_n} \approx \frac{x_n^2 - x_{n-1}x_{n+1}}{2x_n - x_{n+1} - x_{n-1}}$$

Abbiamo ottenuto una nuova approssimazione. In pratica il metodo di accelerazione si utilizza nel seguente modo:

- Si parte con x_0 e si generano x_1 e x_2 con il metodo di Newton.
- Con questi 3 punti si puó costruire $\overline{x_1}$.
- Si pone $x_0 = \overline{x_1}$ e si generano x_1 e x_2 e calcoliamo la nuova approssimazione $\overline{x_2}$.
- Il procedimento si ripete iterativamente.

Si puó dimostrare che questo metodo converge quadraticamente alla soluzione.

3.9 Problema della convergenza nei metodi iterativi

Noi abbiamo una procedura di tipo iterativo e vogliamo vedere sotto quali condizioni questa converge ad una radice.

$$x_{n+1} = T(x_n)$$

Dove T é la funzione di iterazione caratteristica del metodo esaminato. In particolare, nel metodo di Newton $T(x) = x - \frac{f(x)}{f'(x)}$ e noi cerchiamo $\overline{x}$ tale che $f(\overline{x}) = 0$. Affinché x_n converga a $\overline{x}$ T deve essere tale che $T(\overline{x}) = \overline{x}$, cioé, una volta raggiunta la soluzione, non ci si deve piú spostare. $\overline{x}$ é detto *punto fisso*

Teorema 1 (Convergenza di procedure iterative) *Sia $x_{n+1} = T(x_n)$ un metodo iterativo per risolvere $f(x) = 0$.*
Se $\exists \delta > 0$ tale che $\forall x, y \in \underbrace{]\overline{x} - \delta, \overline{x} + \delta[}_{I_\delta}$ risulta

$$|T(x) - T(y)| \leq L\,|x - y|\,,\ \text{con } L < 1$$

se $x_0 \in I_\delta$, allora $x_n \to \overline{x}$ per $n \to \infty$.
Inoltre non esistono altri punti fissi di T in I_δ

Dimostrazione. Se $x_0 \in I_\delta$ allora $|x_1 - \overline{x}| \leq L\,|x_0 - \overline{x}|$ ragionando per induzione si dimostra che:

$$|x_i - \overline{x}| = |T(x_{i-1}) - T(\overline{x})| \leq \cdots \leq L^i |x_0 - \overline{x}|$$

ma poiché $L < 1$, $L^i |x_0 - \overline{x}| \to 0$ per $i \to \infty$. La convergenza é dimostrata.
Supponiamo che $\exists \hat{x} \neq \bar{x}$ tale che $\hat{x} \in I_\delta$ e $\hat{x} = T(\hat{x})$, allora

$$|\hat{x} - \bar{x}| = |T(\hat{x}) - T(\bar{x})| \leq L|\hat{x} - \bar{x}| < |\hat{x} - \bar{x}| \text{ assurdo.}$$

Si é dimostrata anche l'unicitá.

3.10 Metodi quasi Newtoniani

Metodo di Newton generico:

$$x_{n+1} = x_n - \frac{f(x_n}{f'(x_n)}$$

il problema è che f' è difficile e costosa da calcolare e quindi i metodi *quasi Newtoniani* approssimano tale quantità in maniera opportuna.

3.10.1 Metodo delle corde

Sugerimento: se ci roviamo abbastanza vicino alla soluzione $\overline{x}$ allora la pendenza della retta tangente il grafico della funzione non cambia di molto e quindi f' ci dice in qualche modo in che direzione dobbiamo guardare per trovare la soluzione. Quindi il metodo delle corde, che si basa su questo suggerimento, calcola $f'(x_0)$ e lascia invariato questo valore usandolo al posto di $f'(x_i) \quad i = 1 \ldots$. Quindi il metodo diventa

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}$$

Il costo del metodo delle corde è assimilabile a quello di bisezione perché ad ogni passo l'operazione predominante è una valutazione della f; il metodo è convergente di ordine 1, comunque scegliendo il punto di innesco adeguatamente si ottiene una convergenza più veloce.

3.10.2 Metodo delle secanti

Il metodo delle secanti è un "ibrido" tra il metodo di Newton ordinario e quello delle corde, poiché ad ogni passo né calcola la derivata prima della funzione, né tiene fissato un valore, ma approssima la derivata con la pendenza della retta secante il grafico della funzione passante per le due approssimazioni precedenti. In pratica x_1 viene calcolato normalmente e successivamente il metodo diventa del tipo

$$x_{n+1} = \Phi(x_n, x_{n-1})$$

dove viene effettuata la seguente approssimazione

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

dove il secondo membro non è altro che il coefficiente angolare della retta secante il grafico passante per i punti $(x_n, f(x_n))$ e $(x_{n+1}, f(x_{n+1}))$; il metodo ricapitolando diventa

$$x_{n+1} = x_n - \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

e il metodo ovviamente viene detto a passo multiplo.

$$\text{Ordine di convergenza}= \begin{cases} \frac{1+\sqrt{5}}{2} & \text{radici semplici} \\ 1 & \text{radici multiple} \end{cases}$$

Per la complessità valgono considerazioni analoghe al metodo della corde notando che $f(x_{n-1})$ è nota dal passo precedente e non deve quindi essere ricalcolata.

Capitolo 4

Sistemi di equazioni algebriche

Problema: risolvere $F(\underline{x}) = \underline{0}$ con $F = \Omega \subseteq R^m \to r^m$ dove con $F(\underline{x}) = \underline{0}$ viene inteso il sistema $\begin{cases} f_1(\underline{x}) & = & 0 \\ \cdots & \cdots & \cdots \\ f_m(\underline{x}) & = & 0 \end{cases}$ con $f_i : \Omega \subseteq R^m \leftarrow R$.

Problema iniziale: non esiste l'analogo del metodo di bisezione il quale era definito su un intervallo [a,b] tale che $f(a) \cdot f(b) < 0$, quindi non esistono metodi a convergenza globale; esistono comunque il metodo di Newton e le sue varianti.

$$x_{x+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

e se $x_n \approx \overline{x}$ allora $f(x) = f(x_n) + f'(\overline{x})(\overline{x} - x_n) + \ldots$ questa quantità veniva posta a 0 e si trovava la soluzione; nel caso multidimensionale la situazione è analoga, solo che cambia il ruolo di $f'(x_n)$, quindi il tutto si riscrive in

$$F(\underline{x}) = F(\underline{x}_n) + F'(\underline{x}_n)(\overline{x} - \underline{x}_n) + \ldots$$

dove $F'(\underline{x}_n)$ viene detta matrice Jacobiana della funzione F calcolata in $\underline{x}_n$, o semplicemente $J_F(\underline{x}_n)$.

$$\frac{\partial f_i(x_1, \ldots, x_m)}{\partial x_j} = \lim_{\delta \to \infty} \frac{f(x_1, \ldots, x_{j-1}, x_j + \delta, x_{j+1}, \ldots, x_m) - f(x_1, \ldots, x_m)}{\delta}$$

$\forall i, j = 1 \ldots m$ quindi $J_F(\underline{x}_n)$ è la matrice

$$\begin{pmatrix} \frac{\partial f_1(x_n)}{\partial x_1} & \ldots & \frac{\partial f_1(x_n)}{\partial x_m} \\ \ldots & \ldots & \ldots \\ \frac{\partial f_m(x_n)}{\partial x_1} & \ldots & \frac{\partial f_m(x_n)}{\partial x_m} \end{pmatrix}$$

quidi

$$F(\underline{x}) = F(\underline{x}_n) + J_F(\underline{x}_n)(\underline{x} - \underline{x}_n) + \ldots$$

effettuando un'approssimazione al primo ordine e quindi abbiamo che

$$F(\underline{x}_n) + J_F(\underline{x}_n)(\underline{x} - \underline{x}_n) = 0$$

$$\underline{x}_{n+1} = \underline{x}_n - (J_F(\underline{x}_n))^{-1} F(\underline{x}_n) \quad n = 0, 1, \ldots$$

nel caso unidimensionale avevamo come precondizione che $f' \neq 0$ nel caso multidimensionale invece abbiamo $\det(J_F(\underline{x}_n)) \neq 0$. Se $\tilde{x}$ è una soluzione isolata (localmente unica) di $F(\underline{x}) = 0$ allora $J_F(\tilde{x})$ è non singolare, viceversa se la matrice è singolare allora $\tilde{x}$ è "in buona compagnia"; si nota una certa analogia con il caso unidimensionale nel quale le radici multiple annullano più derivate.

$$\underline{x}_{n+1} = \underline{x}_n - (J_F(\underline{x}_n))^{-1} F(\underline{x}_n) \quad n = 0, 1, \ldots$$

in questa equazione comapre un'operazione terribile, l'inversione di una matrice, quindi cerchiamo di ammorbidire l'impatto con questa matrice

$$J_F(\underline{x}_n)(\underline{x}_{n+1} - \underline{x}_n)) = -F(\underline{x}_n)$$

$$d \left\{ \begin{array}{lcl} J_F(\underline{x}_n)\Delta_n & = & -F(\underline{x}_n) \\ \underline{x}_{n+1} & = & \underline{x}_n + \underline{x}_n \end{array} \right. \tag{4.1}$$

con $\Delta_n = (\underline{x}_{n+1} - \underline{x}_n) \quad n = 0, 1, \ldots$.
Costo per ogni passo: 1 valutazione F, una valutazione J_F, un sistema lineare $(2/3n^3)$, per radici localmente isolate il metodo quando (e se) converge, converge quadraticamente (proprietà di tipo locale).

4.1 Metodi quasi Newtoniani in R^m

4.1.1 Metodo delle corde

Per questo metodo $J_F(\underline{x}_n)$ rimane fisso a $J_F(\underline{x}_0)$ e quindi la fattorizzazione di J_F viene fatta una sola volta, il costo diventa quinid 1 valutazione $F + 2n^2$ *flops* per la risoluzione del sistema lineare (4.1).

4.1.2 Metodo delle secanti

$x_{n+1} = x_n - \frac{f(x_n)}{a_n}$ con $a_n = \frac{f_n - f_{n-1}}{x_n - x_{n-1}}$ **problema:** la divisione tra vettori non è un'operazione definita quindi $\underline{x}_n^{-1}$ non esiste ma $\underline{x}_{n+1} = \underline{x}_n - A_n^{-1} F_n$ con $A_n^{-1} \approx (J_F(\underline{x}_n))^{-1}$

R	R^m
$a_n = \frac{f_n - f_{n-1}}{x_n - x_{n-1}}$	$A_n \approx J_F(\underline{x}_n)$
$a_n(x_n - x_{n-1}) = f_n - f_{n-1}$	$A_n(\underline{x}_n - \underline{x}_{n-1}) = F_n - F_{n-1}$
$a_n$definita univocamente	$A_n$definita ma non unica

dal passo $n-1$ al passo n ci siamo spostati verso $(\underline{x}_{n-1}-\underline{x}_n)$ quindi imponiamo come condizione che

$$A_n\underline{y} = A_{n-1}\underline{y} \quad \forall \underline{y} \perp (\underline{x}_{n-1}-\underline{x}_n) \tag{4.2}$$

quindi imponendo

$$\begin{cases} A_n\underline{y} & = & A_{n-1}\underline{y} \\ (\underline{x}_{n-1}-\underline{x}_n)^T\underline{y} & = & 0 \end{cases}$$

si definisce

$$A_n = A_{n-1} + \frac{(F_n - F_{n-1} - A_{n-1}(\underline{x}_n-\underline{x}_{n-1}))(\underline{x}_n-\underline{x}_{n-1})^T}{||\underline{x}_n-\underline{x}_{n-1}||_2^2}$$

in questo modo si può dimostrare che $A_n(\underline{x}_n-\underline{x}_{n-1}) = F_n - F_{n-1}$ e che $A_n\underline{y} = A_{n-1}\underline{y} \quad \forall \underline{y} \perp (\underline{x}_n-\underline{x}_{n-1})$.
Quanto costa A_n? posto $y_n = \frac{(F_n - F_{n-1} - A_{n-1}(\underline{x}_n-\underline{x}_{n-1}))}{||\underline{x}_n-\underline{x}_{n-1}||_2^2}$ e $z_n^T = (\underline{x}_n-\underline{x}_{n-1})^T$ si ha che $A_n = A_{n-1} + y_n z_n^T$. A_{n-1} è nota dal passo precedente $A_{n-1}(\underline{x}_n-\underline{x}_{n-1})$ costa $\approx 2n^2 flops + 1$ valutazione F_n.

Formula di Sherman-Morrison

$$A_n{}^{-1} = A_{n-1}{}^{-1} - \frac{A_{n-1}{}^{-1} y_n z_n^T A_{n-1}{}^{-1}}{1 + z_n^T A_{n-1}{}^{-1} y_n} \tag{4.3}$$

serve comunque $A_0{}^{-1}$ e questa è l'unica "mosca bianca" ovvero l'unico esempio in cui si usa l'inversa di una matrice.

Indice analitico